Adaptive Random Forests from Online Learning and Testing by Betting

Salim I. Amoukou[*]
J.P. Morgan AI Research

Saumitra Mishra
J.P. Morgan AI Research

Manuela Veloso
J.P. Morgan AI Research

Abstract

Online tree ensembles, notably Adaptive Random Forests (ARF), are standard for learning from non-stationary streams, yet key design choices remain largely heuristic. We revisit ARF from an online learning perspective and show that its design reduces to two questions: how to aggregate an evolving pool of trees, and how to renew that pool. For aggregation, we weight trees with an algorithm that has a dynamic regret guarantee, bounding the ensemble's loss relative to the best time-varying combination of trees in hindsight. For pool updates, we replace detector-driven resets with a multiscale schedule: trees receive different base lifetimes, and challengers are introduced ahead of each tree's renewal deadline. Challengers train alongside incumbents, and anytime-valid tests on their paired losses determine promotion. We show that false promotions are controlled across all comparisons, that a challenger with a persistent advantage is promoted within a bounded delay, and that, under explicit coverage and quality conditions, the ensemble's loss stays close to that of an oracle that restarts learning at every change point. On 33 real-world streams, our Online Random Forest (ORF) reduces aggregate loss relative to native ARF by 32.8% in classification and 26.3% in regression. The same reweighting lowers streaming gradient boosting's native loss on these streams, extending the approach beyond bagging.

1 Introduction

Many applications, from fraud detection to energy forecasting, learn from data streams whose generating distributions evolve over time. This non-stationarity, together with the streaming nature of the data, challenges traditional batch learning and motivates adaptive online algorithms.

In batch learning, tree ensembles such as Random Forests [Breiman, 2001] and XGBoost [Chen and Guestrin, 2016] are strong baselines. Inspired by these methods, the data-stream literature has developed online ensembles by integrating bagging and boosting with concept-drift detectors [Gomes et al., 2017, Bifet et al., 2010, 2009, Bifet and Gavalda, 2007, Minku and Yao, 2012] and heuristics for adding, resetting, or replacing base learners in response to distributional shifts.

Despite their empirical success, existing streaming ensembles remain largely driven by heuristics, with limited theoretical guarantees for the overall system. We revisit these methods through the *learning-with-expert-advice* framework [Littlestone and Warmuth, 1994, Vovk, 1990]. Taking *Adaptive Random Forests* (ARF) [Gomes et al., 2017] as the representative algorithm, we replace its aggregation rule, drift detectors, and model-reset rules with mechanisms from *online learning* [Orabona, 2026] and *testing by betting*. The resulting framework is parameter-free (requiring no forgetting factor or prior knowledge of how the best tree weights in hindsight evolve over time) and detector-free; it adapts to non-stationarity automatically and improves substantially over ARF in our experiments. We also apply the same reweighting to Streaming Gradient Boosted Trees (SGBT), the leading boosting-based method, showing that the benefits extend beyond bagging-based ensembles.

[*]Correspondence to: Salim I. Amoukou <salim.ibrahimamoukou@jpmorgan.com>

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

2 Related works and contributions

Streaming ensembles. Streaming ensembles are widely used for their empirical performance and, as in batch learning, fall into bagging- and boosting-based families. Bagging methods, inspired by random forests [Breiman, 2001], include *Adaptive Random Forests* (ARF) [Gomes et al., 2017, 2018], *Streaming Random Patches* (SRP) [Gomes et al., 2019], and *Self-Optimising K-Nearest Leaves* (SOKNL) [Sun et al., 2022]. ARF and SRP are strong baselines in recent comparative studies [Montiel et al., 2020, Wang et al., 2022, Aspis et al., 2025]; ARF is the standard reference for bagging-based stream learning.

Online boosting originated with OzaBoost [Oza and Russell, 2001] and Online Smooth Boosting [Chen et al., 2012], and was later extended by methods such as AXGB [Montiel et al., 2020] and AdIter [Wang et al., 2022]. Historically, bagging-based ensembles, especially ARF, have outperformed boosting approaches. More recently, *Streaming Gradient Boosted Trees* (SGBT) [Gunasekara et al., 2024, 2025] have shown competitive or better results in some settings. While most work focuses on classification [Read and Žliobaité, 2025], regression methods include ARF [Gomes et al., 2018], SGBT [Gunasekara et al., 2025], and SOKNL [Sun et al., 2022]. In this work we target ARF, the leading bagging-based method and a strong choice for both tasks, and later test whether a detached prediction correction also benefits SGBT, its boosting counterpart.

Drift handling: heuristics vs. theory. A central design question in streaming ensembles is how to respond to concept drift. The dominant paradigm, exemplified by ARF, is heuristic and reactive: each base learner is paired with a drift detector (e.g., ADWIN [Bifet and Gavalda, 2007]), and detected drift triggers the creation or replacement of models. This "detect-then-reset" strategy introduces sensitive hyperparameters that can substantially affect performance [Read and Žliobaité, 2025, Žliobaité et al., 2015].

This heuristic evolution contrasts with the field's theoretical origins in *Learning with Expert Advice* (LwE). At a high level, many streaming ensembles can be interpreted as instances of the *Multiplicative Weights* algorithm [Littlestone and Warmuth, 1994, Blum, 1997]. Early systems also explicitly used meta-weights with pool management [Kolter and Maloof, 2005, 2007], including strategies for adding/removing learners under drift, and came with theoretical guarantees. Recent methods introduced additional heuristic components via bagging and ad hoc reset strategies, culminating in modern algorithms such as ARF and SRP. Parallel developments occurred in online boosting, beginning with the seminal works of Freund and Schapire [1995, 1996] and extending to recent methods such as SGBT [Gunasekara et al., 2025]. Over time, the original theoretical foundations and guarantees underlying these approaches, as in Kolter and Maloof [2005, 2007], have largely receded, as research has prioritized empirical performance through heuristic design choices over formal analysis.

Our contribution. We revisit ARF through the LwE framework and modify two components:

1. **Meta-weighting.** We replace ARF's native aggregation, running-accuracy-weighted voting in classification and median predictions in regression, with an online weighting rule having theoretical guarantees for tracking changing mixtures of trees, making it better suited to non-stationary environments.
2. **Pool updates.** We remove the explicit drift detectors, which react only after a change is detected, in favor of a proactive scheme that (i) generates candidate restarts across multiple time scales using geometric lifetimes [Jun et al., 2017, Gopalan et al., 2007, Woodruff, 2007], and (ii) certifies challenger promotions with anytime-valid statistical tests built by testing by betting [Shafer and Vovk, 2001, Shafer et al., 2011, Hendriks, 2018, Orabona and Jun, 2024].

We call the method that changes only the aggregation ARF-W (ARF with adaptive weighting); our main method, Online Random Forest (ORF), additionally replaces the detector-based lifecycle. The result is an ARF-style ensemble with an end-to-end theoretical performance guarantee that also outperforms the original ARF empirically. The same aggregation further improves SGBT, the leading boosting method.

3 A unifying framework

Our goal is to design an online ensemble for non-stationary data streams that (i) continuously incorporates new observations, (ii) forgets obsolete information, and (iii) admits performance guarantees. We formalize this objective within the framework of *learning with expert advice*, which subsumes many practical streaming ensembles. At each round $t = 1, \ldots, T$, the covariates $\boldsymbol{x}_t$ are observed before the outcome $\boldsymbol{y}_t$:

1. **Expert pool.** A set of $1 \leq n_t \leq N$ experts $\mathcal{M}_t = \{m_{t,1}, \ldots, m_{t,n_t}\}$ is available. Each $m_{t,i}$ produces advice $p_{t,i} = m_{t,i}(\boldsymbol{x}_t)$ upon observing $\boldsymbol{x}_t$. Experts may be static or incremental learners (e.g., Hoeffding Trees [Domingos and Hulten, 2000]), whose internal states evolve over time.
2. **Meta-weights.** The meta-algorithm selects weights $\boldsymbol{w}_t$ in the simplex

$$\Delta_{n_t} := \Big\{ \boldsymbol{w} \in \mathbb{R}_+^{n_t} : \sum_{i=1}^{n_t} w_i = 1 \Big\}.$$

   We denote this procedure by $\mathcal{W}$.
3. **Ensemble prediction.** The ensemble predicts $\widehat{\boldsymbol{y}}_t = \sum_i w_{t,i} p_{t,i}$.
4. **Loss feedback.** After observing $\boldsymbol{y}_t$, define the individual losses $l_{t,i} = l_t(m_{t,i}) = \ell(\boldsymbol{y}_t, p_{t,i})$ and the ensemble loss $\ell_t(\mathbf{w}) = \ell(\boldsymbol{y}_t, \sum_i w_i p_{t,i})$. Write $\bar{\ell}_t(\mathbf{w}) = \langle \mathbf{w}, \boldsymbol{l}_t \rangle$ for the weighted average of individual losses. For convex prediction loss, averaging predictions incurs no more loss than averaging their individual losses: $\ell_t(\mathbf{w}) \leq \bar{\ell}_t(\mathbf{w})$.
5. **Meta-update.** The meta-algorithm $\mathcal{W}$ updates its state from the individual losses $\boldsymbol{l}_t = (l_{t,1}, \ldots, l_{t,n_t})$ and selects the next round's weights.
6. **Pool update (base learning).** After scoring, the base learners train on the current observation according to their training rules. Pool-management decisions may add, remove, or restart models before a subsequent prediction.

We represent the pool by N reusable slots, representing the maximum pool size, one weight coordinate per slot, and assign zero prediction weight to inactive slots. A replacement model occupies the same slot and uses the same weight coordinate. This formulation separates *Pool update (base learning)* (training and managing $\mathcal{M}_t$) from *meta-learning* (choosing $\boldsymbol{w}_t$). Many state-of-the-art algorithms, including ARF, SGBT, and AXGB, can be viewed as specific instantiations of this general framework. They differ primarily in their implementation of four components: the base model and its training, the weight update rule, the trigger for pool updates, and the pool-management mechanism. Table 18 in Appendix J compares these components across forest and boosting methods. We thus address two questions: **(1)** Given a pool of trees, how should we weight their predictions? **(2)** Given a weighting rule, how should we update the pool? In the following sections, we keep ARF as is except for two components, for which we present principled replacements: the weight update rule and the pool-management mechanism.

4 Adaptive meta-weighting

Native ARF classification aggregates tree predictions via weighted majority voting, where each tree's vote is weighted by its running accuracy. This performance-weighted scheme is closely related to Hedge, or exponential weights, with update $w_{t+1,i} \propto w_{t,i} \exp(-\gamma l_{t,i})$ on a fixed set of slots. The learning rate $\gamma > 0$ sets how strongly recent losses move the weights. In non-stationary streams, two key limitations arise. First, the right learning rate depends on the unknown rate of drift. Second, the standard Hedge guarantee compares with the *best single expert in hindsight*, controlling $R_T = \sum_{t=1}^{T} \bar{\ell}_t(\boldsymbol{w}_t) - \min_i \sum_{t=1}^{T} l_{t,i}$. This fixed comparator is weak under concept drift: one expert may be best for $t \leq T/2$ and another for $t > T/2$.

A changing benchmark. Consider a hypothetical allocation $u_t \in \Delta_N$ of weight across the current trees on each round. It may keep the same allocation during a stable period and shift weight as different trees become useful. We score it by $\bar{\ell}_t(u_t)$, the weighted average of those trees' individual losses. *Dynamic regret* measures our extra cumulative weighted loss relative to this changing benchmark:

$$R_T(u) = \sum_{t=1}^{T} \big[\bar{\ell}_t(\mathbf{w}_t) - \bar{\ell}_t(u_t)\big], \qquad u = (u_1, \ldots, u_T).$$

To measure how much the benchmark changes, define its total reallocated weight

$$P_T(u) = \frac{1}{2} \sum_{t=2}^{T} \|u_t - u_{t-1}\|_1.$$

Moving from $(0.8, 0.2)$ to $(0.7, 0.3)$ contributes 0.1; moving all weight from one slot to another contributes one. Our goal is therefore an online algorithm whose regret grows slowly with $P_T(u)$, without knowing this variation in advance.

Adaptive learning for Dynamic EnviRonment (ADER) [Zhang et al., 2018], presented as Algorithm 14.1 in Orabona [2026], provides such a guarantee. Another possibility is *Mirror-Descent Weighting* (MDW): Algorithm 2 of Jacobsen and Cutkosky [2022], combined with the available-slot reduction (Appendix D.1).

Theorem 4.1 (Dynamic regret for changing tree weights)**.** *Use either exact MDW (Appendix D.1) or exact ADER weighting (Appendix D.3), with N slots, horizon $T \geq 1$, and at least one available slot on each round. If $l_{t,i} \in [0, 1]$, then, for a universal constant $c_{\mathrm{DR}} > 0$, simultaneously for every $u_1, \ldots, u_T \in \Delta_N$ supported on the available slots,*

$$R_T(u) \leq \underbrace{\begin{cases} c_{\mathrm{DR}} \sqrt{NT\,[1 + P_T(u)] \log(eT)}, & \text{MDW}, \\ 4\sqrt{NT\,[1 + P_T(u) + \log(4 + \log T)]}, & \text{ADER}. \end{cases}}_{\text{cost of adapting to changing weights}} \tag{1}$$

If the prediction loss is convex on a domain containing the model predictions, then the same upper bounds apply to $\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \sum_{t=1}^{T} \bar{\ell}_t(u_t)$.

Interpretation. The guarantee holds simultaneously for every weight sequence supported on the available slots. In particular, it applies to the *best changing mixture of trees in hindsight*: a sequence $u^\star$ minimizing $\sum_{t=1}^{T} \bar{\ell}_t(u_t)$ over these sequences subject to $P_T(u) \leq P$, whenever this class is nonempty. Here P is the total-reallocation budget. For fixed N, the adaptation cost divided by T tends to zero when $P = o(T)$ and $P = o(T/\log(eT))$ for ADER, MDW, respectively, meaning that the total weight movement grows more slowly than the stream length. Under convexity, the forest's average prediction loss is therefore at most the minimum average weighted individual-tree loss among these mixture sequences, plus a vanishing term.

Tracking regret gives a concrete special case. If $u_t = e_{i_t}$ selects one available slot per round and changes slots m times, then $P_T(u) = m$ and the benchmark's cumulative loss is $\sum_{t=1}^{T} l_{t,i_t}$. For fixed N, the cost per round tends to zero when $m = o(T)$ for ADER and $m = o(T/\log(eT))$ for MDW. Thus, the forest can compete with the best such sequence at vanishing average cost even with an increasing number of switches satisfying these conditions. Replacing a tree within the same slot adds no switch or weight movement; the benchmark uses the slot's current occupant.

5 Detector-free update scheduling

The previous section showed how to weight a given expert pool adaptively. Under non-stationarity this is not enough: the pool itself must also evolve.

Most streaming ensembles rely on explicit *concept-drift detectors* to trigger model resets or retraining. These detectors typically depend on heuristic thresholds, which can lead to instability, delayed reactions to shifts, or sensitivity to transient noise [Read and Žliobaité, 2025, Žliobaité et al., 2015]. ARF, for example, employs two detectors with distinct thresholds: a *warning* threshold to begin training a new model and a *drift* threshold to replace an existing model.

We eliminate explicit threshold-based detectors and introduce a principled, self-regulating mechanism that maintains a diverse and adaptive expert pool. Our design has two components: (i) *geometric scheduling*, which assigns slots different base lifetimes, providing frequent fresh starts at short scales and longer comparisons at larger scales; and (ii) *competitive replacement*, which decides when a scheduled candidate has enough evidence against the incumbent to justify promotion.

5.1 Geometric scheduling (finite slots)

One could instantiate a fresh expert at every round, covering all possible restart times and ensuring that the meta-learner has access to a model trained from each potential change point. However, this would lead to an unboundedly growing expert pool, limiting scalability. Instead, we use a finite set of reusable slots with geometric base lifetimes [Hazan and Seshadhri, 2009, Daniely et al., 2015].

Slots, lifetimes, and availability. Partition the N slots as $[N] = \mathcal{P} \dot{\cup} \mathcal{D}$, with permanent slots $\mathcal{P} \neq \varnothing$ and dynamic slots $\mathcal{D}$. Each slot i holds an incumbent $m_i^{(t)}$ and one weight coordinate. Permanent incumbents never expire and remain available. Assign each dynamic slot a fixed lifetime $L_i \in \mathcal{L} := \{\max\{L_{\min}, 2^k\} : k = 0, \ldots, K-1\}$, where $L_{\min} \geq 2$ is even and $K \geq 1$ is an integer. Each dynamic slot may also hold one nonvoting challenger, giving at most $N + |\mathcal{D}|$ live learners. A pre-outcome availability rule may select which dynamic incumbents may receive weight.

Challenger creation and lifecycle outcomes. Each dynamic slot maintains an *incumbent* tree and may introduce a *challenger* as a candidate replacement. Two mechanisms govern this process: *certification*, which promotes the challenger when sufficient evidence from paired loss observations indicates superior performance, and a *deadline retention check*, which determines whether to retain the incumbent or replace it with a fresh tree if certification has not occurred. Fix a dynamic slot and write $L = L_i$. A fresh incumbent started at c has deadline $c + L$ and creates its challenger at $c + L/2$. All live learners are scored before any update. Before each prediction, certification based on earlier paired observations takes priority over the deadline retention check. Promoting a challenger born at b in round v preserves its state and birth time, sets its deadline to $v + L$, and schedules the next challenger at $\max\{v + 1, b + L/2\}$. At an uncertified deadline e, the retention check either retains the incumbent, up to a maximum of $R_{\max} \geq 0$ retentions since its entry as incumbent, or replaces it with a fresh tree. Retention extends the deadline to $e + L$, discards the current challenger, and starts a new one at $e + 1$. All transitions preserve the slot's optimizer state and comparison counter. Figure 1 summarizes these rules; Appendix A.1 provides the bookkeeping and timing proofs.

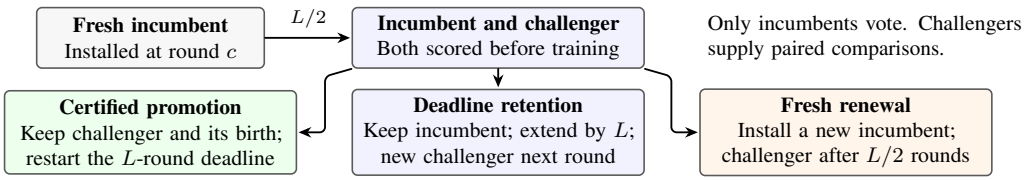


Figure 1: Finite-slot lifecycle for a dynamic slot with base lifetime L. Certification is checked before the deadline retention check; without certification, the retention check determines retention or fresh renewal. Promotion preserves the challenger's training and birth history. All transitions preserve the slot's optimizer state.

Practical instantiation. The evaluated ORF uses 100 slots, four permanent and 96 dynamic, with $K = 48$, $L_{\min} = 50$, and $R_{\max} = 64$. It assigns one to three slots per power, most densely at $2^6 \leq L \leq 2^{17}$, and maintains at most 196 live trees. Base learning follows ARF; MDW supplies the primary weights, with an ADER variant evaluated separately. Appendix G provides the full algorithmic details.

5.2 Competitive replacement via testing by betting

The schedule above regularly creates fresh challengers. We compare each challenger with its incumbent as new outcomes arrive and promote it when enough evidence accumulates. To make this decision without fixing the comparison length in advance, we use *testing by betting* [Grünwald et al., 2020, Shafer, 2024, Choe and Ramdas, 2024]. Growing betting wealth provides evidence against the null that the challenger has no conditional advantage.

Incumbent–challenger advantage and null. Assume losses lie in $[0, 1]$. Index each comparison by its dynamic slot i and a counter $q \geq 1$ that increments whenever a new comparison begins in that slot. Let $c_{i,q}$ denote the challenger created at round $s = s_{i,q}$ and $m_{i,q}$ the incumbent active at that time. Their identities remain fixed throughout the comparison, although both trees may continue training. The comparison terminates if either tree is replaced. On each round $t \geq s$ while the comparison is active, define the *prequential advantage* using losses from predictions made before observing the outcome: $A_t^{(i,q)} := l_t(m_{i,q}) - l_t(c_{i,q}) \in [-1, 1]$. Positive values indicate that the challenger incurs less loss than the incumbent. Let $\mathcal{F}_{t^-}$ contain the information available before the current outcome, including the covariates, model predictions, and observation decision. We test the one-sided conditional null $H_0^{(i,q)} : \mathbb{E}[A_t^{(i,q)} \mid \mathcal{F}_{t^-}] \leq 0$ almost surely on every round: given everything known before the outcome, the challenger is not expected to beat the incumbent on any round. We omit the superscripts when the comparison is clear.

Testing by betting. A bettor starts with wealth $W_{s-1} = 1$ and, at each time $t \geq s$, stakes a fraction $\beta_t \in [0, 1]$ of it on the challenger winning the round. The fraction is chosen before the outcome, using only the information in $\mathcal{F}_{t^-}$ (a *predictable* choice). After observing A_t, the wealth is updated as $W_t = W_{t-1}(1 + \beta_t A_t)$. Under the null, these bets have no positive expected return. Ville's inequality bounds the chance that wealth ever reaches $1/\alpha$ by α: $\mathbb{P}_{H_0^{(i,q)}}(\sup_{t \geq s} W_t \geq 1/\alpha) \leq \alpha$ for any test level $\alpha \in (0, 1)$. We can therefore check the threshold after every observation while retaining this error bound (*anytime validity*). A crossing is evidence against the conditional null. We next show how to adaptively select the betting fractions using an online portfolio strategy with a log-wealth guarantee relative to the best fixed fraction in hindsight, enabling rapid detection of challenger improvements.

Portfolio-selection view. Following Orabona and Jun [2024], we can view the betting game as a portfolio with two assets: one keeps its value, and the other's return follows the challenger's advantage. Their returns are $R_t^{(0)} := 1$ and $R_t^{(1)} := 1 + A_t \in [0, 2]$. Investing a fraction β_t in asset 1 and $1 - \beta_t$ in asset 0 yields the gross return $(1 - \beta_t)R_t^{(0)} + \beta_t R_t^{(1)} = 1 + \beta_t A_t$, which is the one-step wealth multiplier of the betting game. Designing a powerful anytime-valid test therefore reduces to selecting an effective online portfolio strategy.

We use the *Universal Portfolio* (UP), which averages the wealth of all constant betting fractions $\beta \in [0, 1]$ under a Jeffreys prior $F_+ = \mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$ [Cover and Ordentlich, 1996, Orabona and Jun, 2024]. Fractions that have accumulated more wealth have more influence on the next bet. After any $n \geq 1$ paired observations, its log wealth is at most $\log(2\sqrt{n})$ below that of the best constant betting fraction in hindsight (Lemma B.2). Appendix B.1 gives the exact mixture. At level $\alpha_{i,q}$, the comparison *certifies* the challenger at the first threshold crossing, $\tau_{i,q} := \inf\{t \geq s_{i,q} : W_t \geq 1/\alpha_{i,q}\}$. To control errors across the whole stream, we assign each comparison a share of a total error budget α.

Theorem 5.1 (Global false-certification control)**.** *Let $\alpha \in (0, 1)$. Assign prespecified levels $\alpha_a > 0$ to a countable family of distinct, predictably instantiated fixed-pair comparisons, with $\sum_a \alpha_a \leq \alpha$. Under any fixed data-generating law, the probability that any comparison whose conditional null holds almost surely throughout its active lifetime certifies is at most α.*

Allocating comparison levels with total sum at most α controls false certifications across the entire stream (Theorem B.3, Appendix B). For efficiency, we use four equally weighted betting fractions $1/8, 1/4, 1/2, 1$, which retain anytime validity and achieved identical empirical performance to the continuous version. Proposition 6.2 establishes the certification-delay bound for the continuous version.

Is this a drift detector? Our method can be seen as an implicit drift detector, but it differs from classical approaches. Standard detectors flag distributional change via thresholded statistics, which requires careful tuning. We replace arbitrary drift thresholds with structured competition within the model class itself. Our approach is better viewed as evidence-based model selection rather than drift detection.

6 Theoretical guarantees

We now compare the ensemble with an oracle that knows when to restart. Scheduling determines when fresh candidates are created, testing decides when there is evidence to promote them, and weighting determines their contribution to prediction. For analysis, fix deterministic reference boundaries $1 = \tau_1 < \cdots < \tau_{S+1} = T + 1$, with episodes $I_s = [\tau_s : \tau_{s+1} - 1]$ of length L_s. The oracle restarts a fresh tree $m_s^\star$ at each τ_s, predicting each outcome before using it to update that tree.

6.1 Coverage of unknown change times

We first show that the finite-slot lifecycle provides *coverage*: fresh learners are created near any change point and remain alive long enough to accumulate observations. Both a fresh incumbent installed by renewal and a fresh challenger provide such a restart.

Proposition 6.1 (Finite-slot multi-scale coverage)**.** *Follow the lifecycle of Section 5.1, starting from a fresh initialization. Fix a dynamic slot with even base lifetime $L \geq 2$ and any round τ at or after initialization. In a continued execution, the first fresh incumbent or challenger born at or after τ has birth time b satisfying $0 \leq b - \tau \leq L - 1$. The model born at b remains alive and is scored on rounds $b, \ldots, b + L/2 - 1$, possibly changing from challenger to incumbent. The birth and all $L/2$ scoring rounds occur within the finite horizon whenever $\tau + 3L/2 - 2 \leq T$.*

Thus, after any change point, each maintained scale creates a fresh learner within at most $L - 1$ rounds and gives it at least $L/2$ rounds to learn from new observations. This guarantees opportunities for adaptation, but not that the learner will be selected for prediction or achieve good performance, which are addressed in the result below (proof in Appendix A.1). For certification, the lifecycle also ensures regular opportunities to compare challengers against incumbents. New challengers are created at most $3L/2 - 1$ rounds apart, and each receives at least $L/2$ rounds of paired evaluation , unless sufficient evidence allows earlier certification (Proposition A.1).

6.2 Certification delay under persistent advantage

Before contributing to prediction, a challenger must accumulate enough evidence to be *certified*. The next result allows an initial learning period, then assumes a persistent expected advantage over the same paired incumbent.

Proposition 6.2 (Certification delay)**.** *Consider a fixed incumbent–challenger pair using the continuous Jeffreys-mixture test at level $\alpha \in (0, 1)$, and fix $\delta \in (0, 1)$. Suppose the first n_0 paired observations may have arbitrary advantages in $[-1, 1]$, but from that point onward, the challenger has a conditional expected advantage of at least $\Delta \in (0, 1]$ on every round in which the comparison remains active. If the pair remains active long enough, then with probability at least $1 - \delta$, certification occurs within*

$$D_{\mathrm{obs}} = n_0 + O\left(\frac{n_0}{\Delta} + \frac{\log(1/\alpha) + \log(1/\delta) + \log(e/\Delta)}{\Delta^2}\right) \quad \textit{actual paired observations.}$$

The bound captures two intuitive effects. First, smaller advantages require more observations to distinguish from random fluctuations, yielding the Δ^{-2} dependence up to logarithmic factors. Second, unfavorable early observations can delay certification, contributing the n_0/Δ term. Thus, a sustained challenger advantage leads to certification with high probability, provided the comparison remains active long enough. For the continuous-test version of ORF, the lifecycle guarantees sufficient time for certification whenever $D_{\mathrm{obs}} \leq L/2$ and enough rounds remain in the stream.

6.3 End-to-end guarantee

We now connect the lifecycle guarantees to the forest's predictive performance by comparing ORF with an oracle that restarts a learner at each reference boundary. Although ORF cannot restart exactly at these boundaries, its coverage guarantee ensures that fresh learners are created within a bounded delay and given opportunities to adapt. Since they use the same base-learning algorithm on post-change observations, their performance may approach that of the oracle when the regime remains sufficiently stable.

This motivates considering a reference mixture u_t over the currently available trees, whose conditional expected loss is assumed to be within ε_s of the oracle's on certain *useful rounds*. Both the reference weights and useful rounds are determined before observing the current outcome. Theorem 4.1 then bounds the additional loss from learning to track these changing mixtures online.

Theorem 6.3 (End-to-end guarantee)**.** *Use the exact MDW of Theorem 4.1, with convex prediction loss in $[0,1]$ for the forest, its models, and the restarted reference. Let a predictable rule choose weights $u_t \in \Delta_N$ over the currently available models and designate useful rounds before observing the current outcome. Put $u = (u_1, \dots, u_T)$. For fixed $\varepsilon_s \in [0,1]$, assume on useful rounds of episode I_s that $\mathbb{E}[\bar{\ell}_t(u_t) - l_t(m_s^\star) \mid \mathcal{F}_{t^-}] \le \varepsilon_s$. For deterministic bounds $0 \le g_s \le L_s$, suppose that, with probability at least $1 - \delta_{\mathrm{cov}}$, each episode has at most g_s non-useful rounds. Then, for every $\delta_{\mathrm{loss}} \in (0,1)$, with probability at least $1 - \delta_{\mathrm{cov}} - \delta_{\mathrm{loss}}$,*

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \underbrace{\sum_{s=1}^{S} \sum_{t \in I_s} l_t(m_s^\star)}_{L_T^\star} \le \underbrace{\sum_s g_s}_{\text{coverage delay}} + \underbrace{\sum_s (L_s - g_s)\varepsilon_s}_{\text{approximation error}} + \underbrace{c_{\mathrm{DR}} \sqrt{NT\,[1 + P_T(u)] \log(eT)}}_{\text{dynamic-regret cost}} + \underbrace{\sqrt{2T \log(1/\delta_{\mathrm{loss}})}}_{\text{loss fluctuations}}. \tag{2}$$

For exact ADER weighting, the same guarantee holds with the dynamic-regret cost replaced by $4\sqrt{NT[1 + P_T(u) + \log(4 + \log T)]}$.

The bound separates four sources of additional loss relative to the oracle: rounds without a sufficiently competitive tree, the approximation error on useful rounds, the cost of tracking changing mixture weights, and random loss fluctuations. Thus, when the lifecycle provides timely access to well-adapted trees, the forest can approach the oracle's performance, with the remaining gap controlled by the approximation error and dynamic-regret cost (proof in Appendix E).

7 Experiments

Setup. We compare four forests, all built on River's ARF implementation [Montiel et al., 2021] with 100 trees. **Native ARF100 (ARF100-N)** is the primary baseline: ARF's lifecycle with River's default aggregation, running-accuracy-weighted voting in classification and the median in regression. **Uniform ARF100 (ARF100-U)** replaces that aggregation by the arithmetic mean and serves as the weighting control. **ARF-W** keeps ARF's detector-based lifecycle and replaces the aggregation by our adaptive weighter (MDW, Appendix D.1). **ORF** uses the same weighter and additionally replaces the lifecycle by geometric renewal with challengers and deadline retention checks (Appendix I.1). We evaluate on **33 real-world streams**, 17 classification and 16 regression, with seeds 1–3 and identical scored rows across methods. The primary losses are half-Brier, $\frac{1}{2}\sum_c (p_c - y_c)^2$ with one-hot y, and tail-robust MAE in original target units, which excludes each run's largest 0.1% of absolute errors. Appendix I gives preprocessing, horizons, and secondary metrics; Appendix I.3 adds SRP, OzaBoost, OSB, SOKNL, and ARF10.

7.1 Main results

Table 1 reports results for every stream. We aggregate performance using the geometric mean of per-stream loss ratios relative to native ARF100, $G = \exp\big(\frac{1}{D}\sum_{d=1}^{D} \log(\bar{L}_{M,d}/\bar{L}_{\mathrm{ARF100-N},d})\big)$, where $\bar{L}$ denotes the loss averaged over seeds and D is the number of streams. This gives equal weight to each stream, with $G < 1$ indicating an improvement over the baseline. Three findings stand out.

Adaptive weighting alone gives most of the gain. ARF-W, which changes only the aggregation, reduces geometric-mean half-Brier by **25.9%** and tail-robust MAE by **21.1%** relative to native ARF100, and by 27.1% and 29.8% relative to uniform ARF100. The second comparison isolates the weighter from the lifecycle, since ARF-W and ARF differ only in how they combine the same trees.

Scheduled renewal adds a further gain. ORF reduces the losses by **32.8%** and **26.3%** relative to native ARF100, an improvement of 9.4% and 6.6% over ARF-W. Compared with ARF-W, ORF achieves lower mean loss on 15/16 regression streams and 9/17 classification streams, with ARF-W showing its largest advantages on Covtype and the three INSECTS streams.

Table 1: **Primary results with native (N) and uniform (U) ARF100.** Mean ± sample SD over three seeds; lower is better. Bold marks the minimum among all four methods, not significance. GM rows give geometric-mean loss ratios to native ARF100.

Dataset	ARF100-N	ARF100-U	ARF-W	ORF
Classification: half-Brier $\times 10^3$				
Bank Marketing	81.89 ± 0.08	81.90 ± 0.08	$\mathbf{81.39} \pm 0.12$	81.50 ± 0.22
Covtype	103.0 ± 0.2	105.3 ± 0.3	$\mathbf{71.40} \pm 0.36$	74.61 ± 0.42
EEG Eye State	17.00 ± 0.18	17.19 ± 0.19	7.691 ± 0.332	$\mathbf{6.090} \pm 0.112$
Elec2	80.83 ± 0.27	81.38 ± 0.23	55.04 ± 0.36	$\mathbf{53.35} \pm 0.13$
Gas Sensor Drift	141.0 ± 2.4	147.5 ± 2.3	87.49 ± 1.93	$\mathbf{52.21} \pm 0.73$
Hotel Bookings	120.1 ± 0.3	121.0 ± 0.4	90.53 ± 0.02	$\mathbf{69.87} \pm 0.18$
HTTP-KDD99	$\mathbf{0.005} \pm 0.000$	$\mathbf{0.005} \pm 0.000$	$\mathbf{0.005} \pm 0.000$	$\mathbf{0.005} \pm 0.000$
Image Segments	156.1 ± 3.4	162.0 ± 2.8	109.3 ± 3.4	$\mathbf{88.84} \pm 4.47$
INSECTS abrupt	185.3 ± 0.8	186.8 ± 0.8	$\mathbf{175.9} \pm 0.5$	176.6 ± 1.1
INSECTS gradual	163.6 ± 0.5	165.1 ± 0.5	$\mathbf{151.7} \pm 0.8$	153.4 ± 0.9
INSECTS recurring	159.9 ± 0.1	161.0 ± 0.1	$\mathbf{152.3} \pm 0.3$	157.6 ± 0.6
NOAA Weather	145.6 ± 0.3	145.7 ± 0.3	$\mathbf{144.8} \pm 0.1$	145.2 ± 0.5
Occupancy Detection	9.667 ± 0.104	9.700 ± 0.110	6.377 ± 0.052	$\mathbf{5.877} \pm 0.102$
Phishing	73.71 ± 0.35	74.18 ± 0.32	$\mathbf{70.77} \pm 0.73$	71.71 ± 0.84
Rialto	195.7 ± 0.2	213.1 ± 0.9	147.2 ± 0.1	$\mathbf{126.9} \pm 0.3$
Room Occupancy	11.88 ± 0.08	12.08 ± 0.07	4.781 ± 0.292	$\mathbf{3.843} \pm 0.085$
Spam	85.15 ± 0.39	86.72 ± 0.44	46.86 ± 0.77	$\mathbf{43.05} \pm 0.81$
GM ratio to native	1.000	1.017	0.741	0.672
Regression: tail-robust MAE				
Abalone	1.403 ± 0.006	1.396 ± 0.005	1.315 ± 0.007	$\mathbf{1.285} \pm 0.002$
Air Quality CO	0.3697 ± 0.0005	0.3825 ± 0.0005	0.3146 ± 0.0012	$\mathbf{0.3040} \pm 0.0020$
Appliances Energy	44.88 ± 0.01	53.93 ± 0.06	33.61 ± 0.06	$\mathbf{33.18} \pm 0.09$
Beijing PM2.5	45.85 ± 0.12	47.37 ± 0.07	27.48 ± 0.07	$\mathbf{26.06} \pm 0.06$
Bike	76.76 ± 0.27	81.92 ± 0.21	65.56 ± 0.20	$\mathbf{61.15} \pm 0.66$
Demand	21.77 ± 0.08	22.53 ± 0.10	20.75 ± 0.12	$\mathbf{19.50} \pm 0.03$
Gas Turbine	3.259 ± 0.007	3.323 ± 0.006	2.509 ± 0.006	$\mathbf{2.412} \pm 0.005$
House8L	$\mathbf{16794} \pm 51$	17738 ± 53	17564 ± 122	17601 ± 35
Household Power	0.3598 ± 0.0021	0.4097 ± 0.0021	0.1592 ± 0.0007	$\mathbf{0.1484} \pm 0.0006$
Metro	1576 ± 4	1545 ± 4	1435 ± 5	$\mathbf{1281} \pm 4$
NZEnergy	2.005 ± 0.056	2.988 ± 0.088	1.035 ± 0.006	$\mathbf{0.7075} \pm 0.0055$
Online News	$\mathbf{2473} \pm 1$	3221 ± 8	3210 ± 13	2965 ± 12
Seoul Bike	244.1 ± 3.0	259.3 ± 2.1	193.9 ± 1.0	$\mathbf{192.2} \pm 1.9$
Steel Energy	3.411 ± 0.076	4.821 ± 0.115	2.683 ± 0.037	$\mathbf{2.506} \pm 0.043$
Tetouan Power	1626 ± 14	2091 ± 13	992.8 ± 1.1	$\mathbf{991.4} \pm 8.8$
Trump Approval	0.3068 ± 0.0009	0.3157 ± 0.0020	0.2867 ± 0.0021	$\mathbf{0.2642} \pm 0.0084$
GM ratio to native	1.000	1.125	0.789	0.737

SD describes seed-to-seed variation, not a confidence interval for a paired difference; zero SD is at displayed precision.

Both methods beat native ARF100 almost everywhere. They have lower means on 16/17 classification streams (one tie) and 14/16 regression streams. Native ARF100 remains best on House8L and Online News. Appendix K shows reduced loss after abrupt and recurring synthetic shifts, alongside a stationary example favoring native ARF100.

7.2 Ablations

Appendix I.4 isolates each component of ORF on the same 33 streams. Removing the weighter and averaging ORF's trees uniformly raises loss by 53% in classification and 47% in regression, so weighting is the dominant factor. On the same ORF trees, the evaluated ADER variant and MDW give similar aggregate primary losses: ADER/MDW geometric-mean ratios are approximately 1.002 (half-Brier) and 1.006 (tail-robust MAE; Table 13). Of ORF's improvement over ARF-W, the geometric schedule alone captures most; certified promotion adds under 1%, and deadline retention checks add about 1.2%/0.5% in classification/regression. Replacing the retention check's evidence by random deadline retention at the same rate costs about 0.5%. Under uniform voting, ORF's loss is 1.2% higher in classification and 3.7% lower in regression than ARF's.

Table 2: **Resource comparison under the same load, relative to ARF-W (=1).** Geometric means over the 17 classification and 16 regression streams of three-seed mean process CPU and peak-memory ratios; lower is cheaper. Implementations and tree counts differ.

	Classification		Regression	
Method	CPU	Memory	CPU	Memory
ARF100-N (native)	0.82	0.83	1.37	1.03
ARF100-U (uniform)	1.01	0.84	1.37	1.05
ORF	1.36	1.91	1.41	1.58

7.3 Computational cost

Table 2 reports process CPU time and peak memory under the same load on all 33 streams, relative to ARF-W. CPU time is reported rather than elapsed time because runs shared a machine; the implementations are single-threaded, so the two coincide on a dedicated core, and CPU time was stable to 2% across repeats where elapsed time was not (Appendix I). ARF-W costs about the same as native ARF100 in classification and less in regression. ORF's schedule costs $1.4\times$ ARF-W's CPU and 1.6–$1.9\times$ its peak memory; on the full Covtype stream (581,012 rows) the ratio reaches $2.76\times$ CPU with a worst-seed peak of 6.7 GB against 1.4 GB.

8 Application to streaming gradient boosting

The same online-learning approach also applies to boosting. SGBT and its regression counterpart SGBR [Gunasekara et al., 2024, 2025] combine boosting with bagging, each stage averaging a bag of trees and later stages learning from the errors of earlier ones, and the trees are then combined by a fixed additive rule. We test whether a learned per-tree reweighting helps here as it helps the forest. A *detached correction head* adjusts the native booster's output without changing its training. Before the label is observed, it reads each tree's contribution $\phi_{t,i}$ and assigns an adjustment $u_{t,i} \in [-\rho, \rho]$, which can be positive or negative. ADER learns these adjustments, starting from zero; $u_t = 0$ reproduces the native prediction. In regression, the candidate is $\sum_i (1 + u_{t,i})\phi_{t,i}$, with a safeguard limiting the correction based on past errors; in classification, the learned adjustments to individual tree contributions are summed separately for each class and added to the booster's log-probability scores, which are then converted back into valid class probabilities. No new trees are trained. Appendix F gives the head's regret bound for classification log loss and unguarded regression half-squared loss; Appendices H and I.6 give the algorithms and protocol.

Results. We report SGBT's default 5×20 configuration, five stages of twenty bagged trees, which matches the forests' 100 trees. Across the 33 streams, the correction lowers the booster's aggregate loss to 0.703 (half-Brier) and 0.889 (tail-robust MAE) of its native value. The corrected booster is also a strong competitor to the forests: ORF's half-Brier is 0.876 of its value, whereas in regression it beats both forests, with ARF-W and ORF at 1.153 and 1.076 times its tail-robust MAE. In cost, it uses $0.19\times$ ARF-W's CPU in regression but $0.97\times$ in classification, where multiclass streams train one committee per class, with $3.0\times$ and $8.8\times$ ARF-W's peak memory, respectively.

9 Conclusion

We revisited Adaptive Random Forests through the lens of online learning with expert advice, focusing on adaptive tree weighting and pool renewal. ARF-W replaces native aggregation with online weighting algorithms equipped with dynamic-regret guarantees, while ORF further introduces geometric renewal and evidence-guided challenger comparisons. Across 33 real-world streams, both methods improve aggregate loss relative to native ARF100. The uniform ARF100 control and same-tree ablations show that adaptive weighting delivers the largest gains, while geometric renewal accounts for most of ORF's additional improvement, with smaller contributions from promotion and retention. Adaptive correction heads extend the approach to boosted trees. Future work could extend boosting beyond reweighting to include scheduled renewal, and explore hybrid architectures combining drift-triggered replacement, deterministic renewal, and permanent tree slots.

DISCLAIMER

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co., and its affiliates ("JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation, and warranty whatsoever, and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

© 2026 JP Morgan Chase & Co. All rights reserved.

References

Miguel Aspis, Sebastián A. Cajas Ordoñez, Andrés L. Suárez-Cetrulo, and Ricardo Simón Carbajo. Driftmoe: A mixture of experts approach to handle concept drifts. *arXiv preprint arXiv:2507.18464*, 2025.

Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, pages 443–448, 2007.

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. New ensemble methods for evolving data streams. In *ACM SIGKDD*, pages 139–148, 2009.

Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *ECML PKDD*, pages 135–150, 2010.

Avrim Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An online boosting algorithm with theoretical justifications. arXiv:1206.6422, 2012.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

Yo Joong Choe and Aaditya Ramdas. Comparing sequential forecasters. *Operations Research*, 72(4):1368–1387, 2024.

Thomas M. Cover and Erik Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363, 1996. doi: 10.1109/18.485708.

Ashok Cutkosky. Parameter-free, dynamic, and strongly-adaptive online learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2250–2259, 2020. URL https://proceedings.mlr.press/v119/cutkosky20a.html.

Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in Banach spaces. In *Proceedings of the 31st Conference on Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1493–1529, 2018. URL https://proceedings.mlr.press/v75/cutkosky18a.html.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1405–1411, 2015. URL https://proceedings.mlr.press/v37/daniely15.html.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *ACM SIGKDD*, pages 71–80, 2000.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9–10):1469–1495, 2017. doi: 10.1007/s10994-017-5642-8.

Heitor Murilo Gomes, Jean Paul Barddal, Luis Eduardo Boiko, and Albert Bifet. Adaptive random forests for data stream regression. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2018.

Heitor Murilo Gomes, Jesse Read, and Albert Bifet. Streaming random patches for evolving data stream classification. In *IEEE International Conference on Data Mining*, pages 240–249, 2019.

Heitor Murilo Gomes, Anton Lee, Nuwan Gunasekara, Yibin Sun, Guilherme Weigert Cassales, Justin Jia Liu, Marco Heyden, Vitor Cerqueira, Maroua Bahri, Yun Sing Koh, Bernhard Pfahringer, and Albert Bifet. CapyMOA: Efficient machine learning for data streams in python. arXiv:2502.07432, 2025.

Parikshit Gopalan, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. Proceedings of SODA, 2007.

Peter Grünwald, Rianne de Heide, and Wouter M. Koolen. Safe testing. In *Information Theory and Applications Workshop*, 2020.

Nuwan Gunasekara, Bernhard Pfahringer, Heitor Gomes, and Albert Bifet. Gradient boosted trees for evolving data streams. *Machine Learning*, 113(5):3325–3352, 2024.

Nuwan Gunasekara, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Gradient boosted bagging for evolving data stream regression. *Data Mining and Knowledge Discovery*, 39, 2025.

Elad Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th International Conference on Machine Learning*, pages 393–400, 2009. doi: 10.1145/1553374.1553425. URL https://icml.cc/Conferences/2009/papers/75.pdf.

Harrie Hendriks. Test martingales for bounded random variables. arXiv:1801.09418, 2018. URL https://arxiv.org/abs/1801.09418.

Andrew Jacobsen and Ashok Cutkosky. Parameter-free mirror descent. In *Proceedings of the 35th Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 4160–4211, 2022.

Kwang-Sung Jun, Francesco Orabona, Stephen Wright, and Rebecca Willett. Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951, 2017.

J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.

Jeremy Z. Kolter and Marcus A. Maloof. Using additive expert ensembles to cope with concept drift. In *International Conference on Machine Learning*, pages 449–456, 2005.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Leandro L. Minku and Xin Yao. Ddd: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012.

Jacob Montiel, Rory Mitchell, Eibe Frank, Bernhard Pfahringer, Talel Abdessalem, and Albert Bifet. Adaptive xgboost for evolving data streams. In *International Joint Conference on Neural Networks*, 2020.

Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: Machine learning for streaming data in Python. *Journal of Machine Learning Research*, 22(110):1–8, 2021.

Francesco Orabona. Online learning: A modern introduction using convex optimization. arXiv:1912.13213v10, 2026. URL https://arxiv.org/abs/1912.13213v10. Version 10, June 21; Algorithm 14.1 and Theorem 14.4.

Francesco Orabona and Kwang-Sung Jun. Tight concentrations and confidence sequences from the regret of universal portfolio. *IEEE Transactions on Information Theory*, 70(1):436–455, 2024. URL https://arxiv.org/abs/2110.14099v3. Proof locator uses arXiv:2110.14099v3.

Nikunj C. Oza and Stuart J. Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236, 2001.

Jesse Read and Indré Žliobaité. Supervised learning from data streams: An overview and update. *ACM Computing Surveys*, 2025.

Glenn Shafer. Improving data analysis by testing by betting: Optional continuation and descriptive statistics. *The New England Journal of Statistics in Data Science*, 2(2):215–228, 2024.

Glenn Shafer and Vladimir Vovk. *Probability and Finance: It's Only a Game!* John Wiley and Sons, 2001.

Glenn Shafer, Alexander Shen, Nikolai Vereshchagin, and Vladimir Vovk. Test martingales, Bayes factors and p-values. *Statistical Science*, 26(1):84–101, 2011. doi: 10.1214/10-STS347.

Yibin Sun, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Soknl: A novel way of integrating k-nearest neighbours with adaptive random forest regression for data streams. *Data Mining and Knowledge Discovery*, 36(5):2006–2032, 2022.

Vladimir Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 371–386, 1990.

Kun Wang, Jie Lu, Anjin Liu, Yiliao Song, Li Xiong, and Guangquan Zhang. Elastic gradient boosting decision tree with adaptive iterations for concept drift adaptation. *Neurocomputing*, 491:288–304, 2022.

Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting, 2022. URL https://arxiv.org/abs/2010.09686v7. Version 7.

David Paul Woodruff. *Efficient and Private Distance Approximation in the Communication and Streaming Models*. PhD thesis, Massachusetts Institute of Technology, 2007.

Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Indré Žliobaité, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, 98(3): 455–482, 2015.

A Online protocol, scheduling, and notation

All round indices and model lifetimes are integers. We use nested information sets $\mathcal{F}_{t-1} \subseteq \mathcal{F}_{t^-} \subseteq \mathcal{F}_t$. The pre-outcome information $\mathcal{F}_{t^-}$ contains the covariates, current model states and predictions, and the weighting, availability, and retention decisions. The outcome and subsequent updates belong to $\mathcal{F}_t$. Models predict before learning from the current observation. Evidence from outcome t changes the pool no earlier than the next prediction. Losses in the bounded-loss results belong to $[0, 1]$.

A physical model is an instantiated learner whose state evolves through training; replacement creates a different physical model. A reusable slot is a coordinate for aggregation. The available-slot weighting in Appendix D allows an implementation to restrict that pool before each outcome. Let $[N] = \mathcal{P} \dot{\cup} \mathcal{D}$ partition the permanent and dynamic slots, with $\mathcal{P} \neq \varnothing$. Permanent slots never expire and are always available; the evaluated dynamic-slot gate is specified in Appendix G. Comparator weights refer to the current physical occupants of the slots.

A.1 Finite-slot lifecycle and paired evaluation

Fix an even minimum lifetime $L_{\min} \geq 2$ and integers $K \geq 1$ and $R_{\max} \geq 0$, the latter bounding retentions per incumbent. Assign each dynamic slot i a fixed base lifetime $L_i \in \{\max\{L_{\min}, 2^k\} : k = 0, \ldots, K-1\}$. Each dynamic slot starts from a fresh installation, and one lifecycle pass precedes each consecutive scored round. For one such slot, write $L = L_i$ and let a, H, and e denote the incumbent's physical birth time, cumulative lifetime, and deadline; the slot also records its extension count. Here H denotes cumulative lifetime; the optimizer horizon is defined separately in Appendix D. A fresh installation at c sets $a = c$, $H = L$, $e = c + L$, and extension count zero. In each pass, birth is checked before certification and deadline actions: when no challenger is active and $t \geq a + H/2$, one fresh challenger is created. At most one challenger is active per slot.

A certified challenger is promoted before any retention check at the deadline in that pass. If it was born at b and promoted at u, it retains physical birth b and its accumulated score history, and receives $H = L$, deadline $u + L$, and extension count zero. A retention at deadline e preserves the incumbent, increases H by L, sets the new deadline to $e + L$, and discards the challenger. The next challenger is born on the following pass, at $e + 1$. At a deadline, a failed retention condition or $R_{\max}$ completed retentions triggers a fresh incumbent installation. All three transitions clear the challenger-active flag and preserve the slot's comparison counter and aggregation state. On each processed round, every live learner is scored on its cached pre-outcome prediction before training on the current observation, including when it is unavailable for voting or receives no training update. Every active pair therefore supplies a paired observation on each round. Its only termination is its own certified promotion or the scheduled deadline.

The evaluated ORF configuration, with evidence-based retention and L-round extensions, uses $L_{\min} = 50$ and $R_{\max} = 64$; Appendix G gives its slot allocation and implementation.

The following proposition records the timing implied by these lifecycle transitions.

Proposition A.1 (Finite-slot challenger births and paired evaluation)**.** *Under these finite-slot transitions, a fresh installation at c in a slot of even base lifetime $L \geq 2$ first creates a challenger at $c + L/2$. Every challenger born at b, with scheduled deadline e, has budget D for paired observations:*

$$\frac{L}{2} \leq D = e - b \leq L - 1, \qquad b_{j+1} - b_j \leq \frac{3L}{2} - 1$$

for successive challenger births b_j in a continued execution. Every processed round $b, \ldots, \min\{e - 1, T\}$ supplies a paired observation unless that comparison certifies earlier.

Consequently, after any $\tau \geq c$, a new challenger birth occurs within $3L/2 - 1$ rounds whenever that budget lies inside the horizon.

Proof of Proposition A.1. Let a be an incumbent's physical birth, u its entry time as incumbent, and r its completed retentions since entry. The transitions give

$$a \leq u, \qquad H = (r + 1)L, \qquad e = u + (r + 1)L.$$

For a fresh installation at c, the first birth is $c + L/2$ and the deadline is $c + L$, so its comparison budget is $D = L/2$. After a retention at $e = u + (r + 1)L$, the next birth threshold is

$$a + (r + 2)L/2 \leq u + (r + 1)L = e.$$

The birth check has already occurred in pass e, so the next challenger is born at $e+1$. Its deadline is $e+L$, giving budget $L-1$.

If a challenger born at b is promoted at v, at least one paired observation has occurred, hence $b \le v-1$. Its next challenger is born at

$$b' = \max\{v+1, b+L/2\},$$

with deadline $v+L$. If $b' = v+1$, the new budget is $L-1$. Otherwise it is $(v-b)+L/2 \ge L/2+1$; since $b' \ge v+1$, it is again at most $L-1$. Induction over fresh installation, retention, and promotion proves $L/2 \le D \le L-1$ for every challenger. The per-round scoring rule supplies each observation on $b, \dots, \min\{e-1, T\}$ unless the comparison succeeds earlier. If the deadline pass occurs, certification from round $e-1$ is acted on before the retention check at the deadline.

Now take a challenger born at b with deadline e and $D = e-b$. Promotion at $v \le e$ yields next birth $\max\{v+1, b+L/2\}$, a gap at most $\max\{D+1, L/2\} \le L$. Retention yields next birth $e+1$, again a gap at most L. Fresh renewal yields next birth $e+L/2$, a gap at most $D+L/2 \le 3L/2-1$. Together with the first-birth formula, these cases prove the bound on successive births and the stated bound after any $\tau \ge c$. On a finite horizon, that conclusion concerns a realized birth whenever $\tau + 3L/2 - 1 \le T$. □

Proof of Proposition 6.1. We count fresh incumbent installations and challenger creations as births of physical models. Promotion preserves an existing physical model and is not a new birth. We first bound the gaps between these births, then show that each newborn remains alive long enough to receive the stated scores.

A fresh installation at c is followed by a challenger birth at $c+L/2$. After a challenger birth at b, write $D = e-b$ for its comparison budget. Promotion or retention produces the next challenger within L rounds, as proved above. If the deadline instead renews the incumbent, that renewal itself is a fresh birth at e, a gap of $D \le L-1$. These cases exhaust the transitions, so successive fresh physical births are at most L rounds apart. Let b now denote the first such birth at or after the integer scored round τ. If $b = \tau$, the waiting bound is immediate. Otherwise the preceding birth b_- exists because τ is at or after initialization, and $b_- \le \tau - 1$. Therefore

$$b \le b_- + L \le \tau + L - 1.$$

A fresh incumbent born at b has deadline $b+L$, and its first challenger cannot be born before $b+L/2$. Hence no transition removes it before $b+L/2$. A challenger born at b has deadline $e \ge b+L/2$. If it is promoted at $v < b+L/2$, its physical identity is preserved, its new deadline is $v+L > b+L/2$, and its next challenger is born at $\max\{v+1, b+L/2\} \ge b+L/2$. Thus promotion cannot shorten its physical survival below the same bound. In either case, the model is alive on every round $b, \dots, b+L/2-1$. The protocol scores every live learner on each round, including learners unavailable for voting and rounds without training updates. These rounds therefore supply $L/2$ scoring and training opportunities. Finally,

$$b + L/2 - 1 \le \tau + 3L/2 - 2,$$

so the stated horizon condition includes all these observations. □

A.2 Fixed-pair observations and stopping

A comparison retains the physical identities of its incumbent m and challenger c. Both learners may continue training; keeping their identities fixed ties the accumulated evidence to the same comparison. Let $O_t \in \{0,1\}$ be pre-outcome measurable and include whether the comparison is still active. On an observed pair define the paired advantage $B_t = l_t(m) - l_t(c) \in [-1,1]$; otherwise set $B_t = 0$. This is the main-text advantage A_t, with the same zero-on-inactive-round convention. No prediction of an unobserved model is needed on an inactive round. Wealth is one before a predictable start and is frozen after termination. Stopping on a crossing after outcome t includes that factor and sets future factors to one.

The conditional null requires $\mathbb{E}[B_t \mid \mathcal{F}_{t^-}] \le 0$ almost surely at every round. For power, index actual paired observations by j and use their pre-outcome information $\mathcal{G}_{j^-}$, including intervening history. Skipped rounds do not count toward an observation budget.

B Anytime-valid testing preliminaries

Lemma B.1 (Ville's inequality)**.** *If $(E_t)_{t\geq 0}$ is a nonnegative supermartingale with $E_0 = 1$, then, for any test level $\alpha \in (0,1)$, $\Pr(\sup_{t\geq 0} E_t \geq 1/\alpha) \leq \alpha$.*

Proof. For $\tau = \inf\{t : E_t \geq 1/\alpha\}$, bounded optional stopping gives $1 \geq \mathbb{E}[E_{\tau\wedge n}] \geq \alpha^{-1}\Pr(\tau \leq n)$. Let $n \to \infty$. To include paths that approach $1/\alpha$ without attaining it, apply the same argument with threshold $c \in (1, 1/\alpha)$. The displayed supremum event implies a finite crossing of c, so its probability is at most $1/c$. Let $c \uparrow 1/\alpha$. □

B.1 Mixture e-processes and universal portfolios

The asset returns 1 and $1 + B_t$ are nonnegative. A constant fraction $\beta \in [0,1]$ invested in the second asset gives return $1 + \beta B_t$. For any fixed probability distribution Π on $[0,1]$, write

$$W_t(\beta) = \prod_{u=1}^{t}(1+\beta B_u), \qquad E_t^{\Pi} = \int_0^1 W_t(\beta)\, d\Pi(\beta), \qquad E_0^{\Pi} = 1,$$

using the zero-before-start and stopped-increment convention above. The products are nonnegative and bounded by 2^t at finite times. Conditional integration gives

$$\mathbb{E}[E_t^{\Pi} \mid \mathcal{F}_{t^-}] = \int_0^1 W_{t-1}(\beta)\big(1+\beta\mathbb{E}[B_t \mid \mathcal{F}_{t^-}]\big)\, d\Pi(\beta) \leq E_{t-1}^{\Pi}$$

under the conditional null. The tower property gives a supermartingale on the post-outcome filtration. This proves validity for the continuous Jeffreys prior and for every fixed discrete mixture in the same argument.

When $E_{t-1}^{\Pi} > 0$, expansion of the last factor gives the predictable portfolio fraction

$$\beta_t^{\Pi} = \frac{\int_0^1 \beta W_{t-1}(\beta)\, d\Pi(\beta)}{\int_0^1 W_{t-1}(\beta)\, d\Pi(\beta)}, \qquad E_t^{\Pi} = E_{t-1}^{\Pi}(1+\beta_t^{\Pi}B_t). \tag{3}$$

If mixture wealth is zero, component wealth is zero for Π-almost every fraction, so wealth remains zero. For $F_+ = \mathrm{Beta}(\frac{1}{2},\frac{1}{2})$ the denominator is strictly positive at finite times, since every factor is positive for $0 < \beta < 1$.

The next lemma applies the two-asset universal-portfolio wealth comparison to paired advantages [Cover and Ordentlich, 1996, Orabona and Jun, 2024].

Lemma B.2 (Jeffreys-mixture log-wealth comparison)**.** *For $n \geq 1$ and any $B_1, \ldots, B_n \in [-1,1]$,*

$$\log\int_0^1 \prod_{j=1}^{n}(1+\beta B_j)\, dF_+(\beta) \geq \max_{\beta\in[0,1]}\sum_{j=1}^{n}\log(1+\beta B_j) - R_n,$$

where

$$R_n = \log\left(\frac{\sqrt{\pi}\,\Gamma(n+1)}{\Gamma(n+\frac{1}{2})}\right) \leq \log(2\sqrt{n}) = \tfrac{1}{2}\log n + O(1).$$

Proof. Expand $W_n(\beta) = \prod_j((1-\beta)+\beta(1+B_j)) = \sum_{k=0}^{n} c_k\beta^k(1-\beta)^{n-k}$, where $c_k \geq 0$. The coefficientwise Jeffreys inequality is

$$\frac{\max_{\beta\in[0,1]}\beta^k(1-\beta)^{n-k}}{\int_0^1 \beta^k(1-\beta)^{n-k}\, dF_+(\beta)} = \frac{\pi\Gamma(n+1)(k/n)^k(1-k/n)^{n-k}}{\Gamma(k+\frac{1}{2})\Gamma(n-k+\frac{1}{2})} \leq \frac{\sqrt{\pi}\Gamma(n+1)}{\Gamma(n+\frac{1}{2})},$$

with $0^0 = 1$. This is the two-asset universal-portfolio inequality [Cover and Ordentlich, 1996]; see the bound preceding Theorem 9 and its coefficient-ratio proof in Orabona and Jun [2024, arXiv version 3, p. 16]. Multiplying by c_k and summing proves $W_n(\beta) \leq e^{R_n}E_n$ for every β. Taking logarithms proves the first assertion. Finally $e^{R_1} = 2$ and

$$e^{R_{n+1}-R_n} = \frac{n+1}{n+1/2} \leq \sqrt{\frac{n+1}{n}},$$

so induction gives $e^{R_n} \leq 2\sqrt{n}$. The sharper expansion $R_n = \frac{1}{2}\log n + \frac{1}{2}\log\pi + O(n^{-1})$ follows by applying Stirling's formula to the displayed Gamma ratio. □

B.2 Global error spending

The following result applies Ville's inequality (Lemma B.1) and a countable union bound to the allocated test levels.

Theorem B.3 (Global false-certification control)**.** *Let $\mathcal{C}$ be a prespecified countable set of potential comparison indices with levels $\alpha_a \in (0,1)$. Each instantiated comparison has a distinct index, a predictably chosen fixed physical pair, a fixed mixture prior, and the stopping convention of Appendix A. Fix a data-generating law P, and let $\mathcal{C}_0(P)$ contain those indices for which $\mathbb{E}_P[B_t^{(a)} \mid \mathcal{F}_{t-}] \le 0$ almost surely at every round. Then*

$$P\left(\exists a \in \mathcal{C}_0(P) : \sup_t E_t^{(a)} \ge 1/\alpha_a\right) \le \sum_{a \in \mathcal{C}_0(P)} \alpha_a \le \sum_{a \in \mathcal{C}} \alpha_a.$$

Proof. For each $a \in \mathcal{C}_0(P)$, the preceding mixture argument gives a nonnegative supermartingale starting at one, including before instantiation and after termination. Lemma B.1 bounds its crossing probability by α_a. A countable union bound proves the result. The null subfamily is defined under P, not selected from realized signs or future outcomes; comparisons outside it may have positive advantage. Certified promotions require such a crossing, proving Theorem 5.1 as well. □

For ORF, initialize each slot's comparison count at zero and increment it before allocating a new test. The stored level is $\alpha_{i,q} = 0.05/[96q(q+1)]$, $q \ge 1$, and q is retained across physical replacements. Therefore

$$\sum_{i=1}^{96} \sum_{q=1}^{\infty} \alpha_{i,q} = 0.05.$$

The equal four-point prior has component masses $1/4$. Classification tests use half-Brier; regression tests use $\min\{(f-\widehat{y})^2/4, 1\}$, so their paired differences are bounded as required. False certification means crossing under the conditional null; the guarantee concerns neither subsequent model quality nor deadline renewal.

B.3 The retention check

The following elementary implication concerns cumulative losses already observed on the comparison window.

Proposition B.4 (Retrospective evidence-backed retention)**.** *For one fixed incumbent m and challenger c, let $l_j(m), l_j(c) \in [0,1]$ be losses on the same n comparison observations. Set $B_j^{\mathrm{keep}} = l_j(c) - l_j(m)$ and*

$$E_n^{\mathrm{keep}} = \sum_{\beta} p_\beta \prod_{j=1}^{n} (1 + \beta B_j^{\mathrm{keep}}),$$

where $0 < \beta \le 1$, $p_\beta \ge 0$, and $\sum_\beta p_\beta = 1$. Then

$$E_n^{\mathrm{keep}} > 1 \quad \Longrightarrow \quad \sum_{j=1}^{n} l_j(m) < \sum_{j=1}^{n} l_j(c).$$

Proof. Some positive-mass component product exceeds one. Its factors are positive, and for that $\beta > 0$,

$$0 < \sum_j \log(1 + \beta B_j^{\mathrm{keep}}) \le \beta \sum_j B_j^{\mathrm{keep}}.$$

This proves the strict cumulative-loss ordering. □

The evaluated retention rule applies this implication to the late comparison observations and retains only while fewer than $R_{\max}$ extensions have occurred. Threshold one establishes the observed cumulative-loss ordering on that window, with no guarantee of future quality. Promotion uses the significance threshold $1/\alpha$.

C UP evidence accumulation under a positive advantage

Theorem C.1 (Continuous-mixture certification after bounded burn-in)**.** *Consider a fixed pair's continuous Jeffreys mixture in actual paired-observation time, with $B_j \in [-1,1]$ and level $\alpha \in (0,1)$. Fix $n_0 \in \mathbb{Z}_{\geq 0}$, $\Delta \in (0,1]$, and $\delta \in (0,1)$. Before certification, suppose*

$$\mathbb{E}[B_j \mid \mathcal{G}_{j^-}] \geq \Delta \qquad (j > n_0).$$

The first n_0 observations may be arbitrary in $[-1,1]$. Keep the pair active and supply the observation budget below unless it certifies earlier. Define

$$A_\Delta = \log\frac{1}{\alpha} + n_0 \log\frac{1}{1-\Delta/2} + \log\frac{\pi}{(\sqrt{2}-1)\sqrt{\Delta}}, \qquad n = \left\lceil \max\left\{ \frac{32}{\Delta^2}\log\frac{1}{\delta},\ \frac{8A_\Delta}{\Delta^2} \right\} \right\rceil.$$

With probability at least $1-\delta$, certification occurs within $n_0 + n$ actual paired observations. The statement also holds conditionally at a predictable comparison start when its parameters and hypotheses are specified there.

Proof. The proof controls the average advantage after burn-in, then converts it into a wealth lower bound for an interval of small betting fractions. Integrating over that interval yields certification.

If the test certifies before the budget, complete the remaining proof-only sequence by zero during any remaining burn-in indices and by Δ thereafter. This completion is chosen using information available before the next observation, so the completed $\bar{B}_j$ remains bounded and satisfies the post-burn conditional-mean bound. On a path without an earlier crossing, the active-pair and paired-evaluation assumptions ensure every increment through the budget is actual.

Put $X_j = \bar{B}_{n_0+j}$, $1 \leq j \leq n$, and $\mu_j = \mathbb{E}[X_j \mid \mathcal{G}_{(n_0+j)^-}] \geq \Delta$. The conditional range width is at most two, so conditional Hoeffding gives

$$\mathbb{E}[\exp\{\lambda(X_j - \mu_j)\} \mid \mathcal{G}_{(n_0+j)^-}] \leq e^{\lambda^2/2}.$$

Since $\sum_{j=1}^n \mu_j \geq n\Delta$, the event below requires $\sum_{j=1}^n (X_j - \mu_j) < -n\Delta/4$. Iteration over the nested pre- and post-outcome information, followed by exponential Markov with $\lambda = -\Delta/4$, yields

$$\Pr\left(\sum_{j=1}^n X_j < \frac{3n\Delta}{4} \right) \leq e^{-n\Delta^2/32} \leq \delta.$$

On the complementary event, for every $\beta \in [\Delta/4, \Delta/2]$, use $\log(1+x) \geq x - x^2$ for $|x| \leq 1/2$ to obtain

$$\sum_{j=1}^n \log(1+\beta X_j) \geq \beta \sum_{j=1}^n X_j - \beta^2 \sum_{j=1}^n X_j^2 \geq n(3\beta\Delta/4 - \beta^2) \geq n\Delta^2/8.$$

The last quadratic is concave in β and equals $\Delta^2/8$ at both endpoints. Every burn-in factor on this interval is at least $1 - \Delta/2 > 0$, and its Jeffreys mass satisfies

$$F_+([\Delta/4, \Delta/2]) = \frac{1}{\pi}\int_{\Delta/4}^{\Delta/2} \frac{d\beta}{\sqrt{\beta(1-\beta)}} \geq \frac{(\sqrt{2}-1)\sqrt{\Delta}}{\pi}.$$

Integrating only over that interval lower-bounds the completed wealth by

$$\frac{(\sqrt{2}-1)\sqrt{\Delta}}{\pi}(1-\Delta/2)^{n_0}\exp(n\Delta^2/8) \geq 1/\alpha.$$

If no actual crossing has occurred earlier, completed and actual wealth coincide through this budget. Hence actual wealth crosses by the stated observation, proving the probability claim. The same argument can be conditioned at the predictable comparison start. □

Since $-\log(1-\Delta/2) \leq \Delta$ for $0 < \Delta \leq 1$, the additional post-burn budget is

$$O\left(\frac{n_0}{\Delta} + \frac{\log(1/\alpha) + \log(1/\delta) + \log(e/\Delta)}{\Delta^2} \right),$$

which proves Proposition 6.2. The result counts observed pairs: a calendar-time application must fit its paired-evaluation window and the evaluation horizon. The common validity proof in Appendix B.1 also covers the four-point implementation; this quantitative delay calculation specifically uses the continuous Jeffreys prior's mass near Δ.

D Mirror-Descent Weighting (MDW) and the dynamic-regret guarantee

We analyze the exact MDW construction used for forest weighting, then give ADER as an alternative and as the optimizer for the detached boosting heads. The forest guarantees compare with changing allocations of tree weights. Selecting one tree at a time gives the switch-count specialization. All logarithms without a subscript are natural.

D.1 MDW and available-slot feedback

The primary experimental runs use MDW, which combines the dynamic-regret optimizer of Jacobsen and Cutkosky [2022, Algorithm 2] with the available-slot reduction below. Here we specify its ideal recurrence; Algorithm 3 lists the numerical safeguards in the evaluated implementation. Let $a_t \in \{0,1\}^N$ be a pre-outcome availability mask with at least one nonzero coordinate. The coordinates $l_{t,i} \in [0,1]$ are the individual losses of the current physical models, and the optimizer state persists through replacements. Losses on unavailable coordinates may be chosen arbitrarily in $[0,1]$, since the mask removes their contribution. Write $\Delta(a_t) = \{u \in \Delta_N : a_t \odot u = u\}$.

For an integer horizon $H \geq 1$, let $G = \sqrt{N}$, $\epsilon_{\mathrm{JC}} = 1$, and set

$$J_{\mathrm{DR}} = \max\{1, \lceil \log_2 \sqrt{H} \rceil\}, \qquad \eta_k = \min\{2^k/(G\sqrt{H}), 1/G\}, \quad 1 \leq k \leq J_{\mathrm{DR}}.$$

Initialize $v_{1,k} = 0$, allocate $\epsilon_{\mathrm{JC}}/J_{\mathrm{DR}}$ to each component, and define

$$r_t = \sum_k v_{t,k}, \qquad V_t = 4G^2 + \sum_{s<t} \|h_s\|_2^2, \qquad \alpha_t = \frac{(\epsilon_{\mathrm{JC}}/J_{\mathrm{DR}})G^2}{V_t \log^2(V_t/G^2)}.$$

Here α_t is an optimizer scale, distinct from the testing level α. For its unrestricted proposal r_t, first set $z_t = (r_t)_+$, its Euclidean projection onto the nonnegative orthant. Available-slot normalization uses the standard sleeping-expert reduction [Orabona, 2026, Algorithm 12.2 and Theorem 12.13]:

$$\begin{aligned} s_t &= \langle a_t, z_t \rangle, \qquad w_{t,i} = \frac{a_{t,i} z_{t,i}}{s_t} \quad (s_t > 0), \\ b_t &= \langle \mathbf{w}_t, \boldsymbol{l}_t \rangle, \qquad g_t = a_t \odot (\boldsymbol{l}_t - b_t \mathbf{1}). \end{aligned} \tag{4}$$

When $s_t = 0$, use uniform weights over the available coordinates. Put $d_t = z_t - r_t$, and feed

$$h_t = \begin{cases} g_t - \dfrac{\langle g_t, d_t \rangle_+}{\|d_t\|_2^2} d_t, & d_t \neq 0, \\ g_t, & d_t = 0. \end{cases} \tag{5}$$

to the core. Its update is

$$\theta_{t,k} = \begin{cases} \dfrac{2\log(1 + \|v_{t,k}\|_2/\alpha_t)}{\eta_k \|v_{t,k}\|_2} v_{t,k} - h_t, & v_{t,k} \neq 0, \\ -h_t, & v_{t,k} = 0, \end{cases}$$

$$v_{t+1,k} = \alpha_{t+1} \left[\exp\left\{ \frac{\eta_k}{2} (\|\theta_{t,k}\|_2 - 2\eta_k \|h_t\|_2^2)_+ \right\} - 1 \right] \frac{\theta_{t,k}}{\|\theta_{t,k}\|_2}, \tag{6}$$

with value zero when $\theta_{t,k} = 0$. The inverse gradient uses the old α_t and the new point uses α_{t+1}.

The correction in (5) is the nonnegative-orthant case of Cutkosky [2020, Algorithm 1 and Theorem 2], which refines the earlier projection reduction of Cutkosky and Orabona [2018]. The next lemma combines this correction with normalization.

Lemma D.1 (Expert-loss transfer for MDW)**.** *For every $u_t \in \Delta(a_t)$,*

$$b_t - \langle u_t, \boldsymbol{l}_t \rangle \leq \langle h_t, r_t - u_t \rangle, \qquad \|h_t\|_2 \leq \|g_t\|_2 \leq \sqrt{N}. \tag{7}$$

Proof. If $s_t > 0$, then $a_t \odot z_t = s_t \mathbf{w}_t$, so $\langle g_t, z_t \rangle = s_t(\langle \mathbf{w}_t, \boldsymbol{l}_t \rangle - b_t) = 0$. If $s_t = 0$, every available coordinate of z_t is zero, giving the same identity. Also, $a_t \odot u_t = u_t$ and $\sum_i u_{t,i} = 1$ give $\langle g_t, u_t \rangle = \langle \boldsymbol{l}_t, u_t \rangle - b_t$. Hence the excess loss is $\langle g_t, z_t - u_t \rangle$. Since $r_t = z_t - d_t$ and the nonnegative vectors z_t, d_t have disjoint supports, $\langle d_t, r_t \rangle = -\|d_t\|_2^2$. If $\langle g_t, d_t \rangle \leq 0$, then $h_t = g_t$

and $\langle g_t, z_t - u_t\rangle = \langle h_t, r_t - u_t\rangle + \langle g_t, d_t\rangle \le \langle h_t, r_t - u_t\rangle$. Otherwise, put $c = \langle g_t, d_t\rangle / \|d_t\|_2^2 > 0$. Then $h_t = g_t - c d_t$, and

$$\begin{aligned}\langle h_t, r_t - u_t\rangle - \langle g_t, z_t - u_t\rangle &= -\langle g_t, d_t\rangle - c\langle d_t, r_t - u_t\rangle \\ &= c\langle d_t, u_t\rangle \ge 0.\end{aligned}$$

Moreover, $\|h_t\|_2^2 = \|g_t\|_2^2 - \langle g_t, d_t\rangle^2 / \|d_t\|_2^2 \le \|g_t\|_2^2$. Together with $b_t \in [0,1]$ and $|g_{t,i}| \le 1$, this proves both claims. □

For $H \ge 2$, the ideal recurrence, rate grid, initialization, component budgets, and energy accumulator match the source algorithm. The convention $J_{\mathrm{DR}} = 1$ also defines the update for $H = 1$. The MDW numerical safeguards in Algorithm 3 are not part of the real-arithmetic guarantee below.

D.2 Dynamic regret for changing tree weights

MDW part of Theorem 4.1. Use $H = T$ and fix any comparator sequence $u_t \in \Delta(a_t)$. When $T = 1$, the theorem follows from $R_1(u) \le 1$. For $T \ge 2$, put

$$\overline{\Lambda}_T = T^2(T+4)\log^2(T+4) J_{\mathrm{DR}}, \qquad \Gamma_T = \max\left\{1, \log\left(1 + \frac{9}{4}\overline{\Lambda}_T\right)\right\},$$

and write $P_2 = \sum_{t=2}^{T} \|u_t - u_{t-1}\|_2$. For two simplex vectors, $z = u - v$ has zero coordinate sum, so its positive and negative parts both have ℓ_1 norm $\|z\|_1/2$ and

$$\|z\|_2^2 = \|z_+\|_2^2 + \|(-z)_+\|_2^2 \le 2(\|z\|_1/2)^2 = \tfrac{1}{2}\|z\|_1^2.$$

Taking square roots and summing over t gives $P_2 \le \frac{1}{\sqrt{2}} \sum_{t=2}^{T} \|u_t - u_{t-1}\|_1 = \sqrt{2}\, P_T(u)$.

By Lemma D.1, the linear losses $x \mapsto \langle h_t, x\rangle$ have Lipschitz constant at most $G = \sqrt{N}$. Apply Jacobsen and Cutkosky [2022, Theorem 4 and Appendix D.4] with $\epsilon_{\mathrm{JC}} = 1$. Its comparator norm satisfies $M = \max_t \|u_t\|_2 \le 1$, its path length is P_2, and

$$\sum_{t=1}^{T} \|h_t\|_2^2 \|u_t\|_2 \le NT, \qquad 4 + G^{-2} \sum_{t=1}^{T} \|h_t\|_2^2 \le T + 4.$$

Thus its logarithmic factor is at most Γ_T. Summing (7) and substituting these bounds gives

$$\begin{aligned}R_T(u) \le 2\sqrt{N} + 6\sqrt{2NT(1 + \sqrt{2}P_T(u))\Gamma_T} \\ + 4\sqrt{N}(1 + \sqrt{2}P_T(u))\Gamma_T.\end{aligned} \tag{8}$$

This is the source guarantee specialized to the available-slot simplex.

To obtain the simpler main-text form, use $J_{\mathrm{DR}} \le T$, $T + 4 \le 3T$, and $\log^2(3T) \le 3T$ to get $\overline{\Lambda}_T \le 9T^5$ and $\Gamma_T \le 5\log(eT)$. Write $Q = (1 + P_T(u))\log(eT)$. Then $(1 + \sqrt{2}P_T(u))\Gamma_T \le 5\sqrt{2}Q$. If $Q \le T$, (8) is at most

$$\left(2 + 6\sqrt{10\sqrt{2} + 20\sqrt{2}}\right)\sqrt{NTQ} < 60\sqrt{NTQ},$$

where we used $\sqrt{N}\,Q \le \sqrt{NTQ}$ and $TQ \ge 1$. If $Q > T$, both b_t and $\langle u_t, \boldsymbol{l}_t\rangle$ lie in $[0,1]$, so $R_T(u) \le T \le \sqrt{NTQ}$. Hence $c_{\mathrm{DR}} = 60$ suffices for the MDW bound in (1), including $T = 1$. All inequalities hold for every realized feedback sequence and every comparator sequence, so the comparator may be chosen in hindsight; no probabilistic assumption or union bound is needed. This also permits adaptive pre-outcome masks and feedback. For convex prediction loss, $\ell_t(\mathbf{w}_t) \le \bar{\ell}_t(\mathbf{w}_t)$, which gives the prediction-loss statement. The comparator is still scored by its weighted individual losses $\ell_t(u_t)$; the theorem makes no claim against the prediction loss of the comparator mixture itself. □

Best mixture within a budget. For MDW with full availability and any $P \geq 0$, the set of sequences $u \in \Delta_N^T$ with $P_T(u) \leq P$ contains the constant sequences and is a closed subset of the compact set Δ_N^T, because P_T is continuous. The continuous function $\sum_t \bar{\ell}_t(u_t)$ therefore attains its minimum on it, and applying (1) to a minimizer gives

$$\sum_{t=1}^{T} \bar{\ell}_t(\mathbf{w}_t) \leq \min_{u:\, P_T(u) \leq P} \sum_{t=1}^{T} \bar{\ell}_t(u_t) + c_{\mathrm{DR}} \sqrt{NT(1+P)\log(eT)}$$

simultaneously for every P, which is the form used in the main text. For fixed N, this bound gives vanishing average regret whenever $(1+P)\log(eT) = o(T)$; the algorithm does not need to know P. At $P = 0$ the objective is linear in one vector, so a minimizer sits at a vertex: the budget-zero benchmark is the best single slot, following its current occupant, as in the Hedge comparison. With varying availability, restrict to $u_t \in \Delta(a_t)$. These are closed constraints, so the minimum is still attained when the class is nonempty. The class may be empty for small P, and we use $\min \varnothing = +\infty$.

Single-tree specialization. Let $\rho = (i_t)_{t=1}^T$ select one available slot per round, $a_{t,i_t} = 1$, and let $\mathrm{sw}(\rho) = \sum_{t=2}^T \mathbf{1}\{i_t \neq i_{t-1}\}$ count its slot changes. For $u_t = e_{i_t}$,

$$\tfrac{1}{2}\|u_t - u_{t-1}\|_1 = \mathbf{1}\{i_t \neq i_{t-1}\}, \qquad \bar{\ell}_t(u_t) = l_{t,i_t},$$

so $P_T(u) = \mathrm{sw}(\rho)$ and the MDW bound in (1) reads

$$\sum_{t=1}^{T} \bar{\ell}_t(\mathbf{w}_t) - \sum_{t=1}^{T} l_{t,i_t} \leq c_{\mathrm{DR}} \sqrt{NT[1+\mathrm{sw}(\rho)]\log(eT)}.$$

Minimizing over sequences with at most m changes gives the tracking special case of the main text; under convexity the left side may be replaced by $\sum_t \ell_t(\mathbf{w}_t) - \sum_t l_{t,i_t}$. Every such sequence is a mixture sequence with $P_T(u) \leq m$, so the mixture benchmark with budget m is never worse than the best tree sequence with m changes. Replacing a tree while keeping its slot and weight contributes nothing to P_T: the benchmark uses the current occupant. The episode comparison separately requires that its comparator weights and useful-round indicators be determined before the current outcome.

D.3 ADER as an alternative

ADER gives an alternative simplex weighter and is the optimizer used by the detached boosting heads, where the domain is a box. We recall its linear-loss bound before giving the corresponding forest guarantee. Let $\mathcal{K} \subseteq \mathbb{R}^d$ be a nonempty closed convex set of Euclidean diameter at most $D > 0$, let $G > 0$, and supply an integer optimizer horizon $H \geq 1$. The following version combines projected-gradient learners at different rates [Zhang et al., 2018]. We use the version in Orabona [2026, Algorithm 14.1 and Theorem 14.4] and include its linear-loss proof for completeness.

Algorithm 1: ADER on a convex domain

Inputs: domain $\mathcal{K}$, diameter bound D, feedback bound G, horizon H, and initial point $x_0 \in \mathcal{K}$. Set

$$J = 1 + \left\lceil \log_2 \sqrt{4 + 8H} \right\rceil, \qquad \eta_k = \frac{D2^k}{G\sqrt{H}} \ (0 \leq k < J), \qquad \gamma = \frac{\sqrt{2\log J}}{GD\sqrt{H}}.$$

Initialize $x_{1,k} = x_0$ and $\pi_{1,k} = 1/J$ for all k. On each round t:

1. Return the proposal $r_t = \sum_k \pi_{t,k} x_{t,k}$.
2. Receive linear feedback g_t satisfying $\|g_t\|_2 \leq G$. Cache $a_{t,k} = \langle g_t, x_{t,k} \rangle$ using the pre-update points.
3. Update every component and its mixing weight:

$$x_{t+1,k} = \Pi_{\mathcal{K}}(x_{t,k} - \eta_k g_t), \qquad \pi_{t+1,k} = \frac{\pi_{t,k} e^{-\gamma a_{t,k}}}{\sum_j \pi_{t,j} e^{-\gamma a_{t,j}}}. \tag{9}$$

For forests, take $\mathcal{K} = \Delta_N$, $D = \sqrt{2}$, $G = \sqrt{N}$, $H = T$, and $x_0 = \mathbf{1}/N$; apply (4). Here $z_t = r_t$, so the orthant correction is unnecessary and the core receives g_t.

Lemma D.2 (ADER linear-loss guarantee)**.** *For any $1 \le T' \le H$, any feedback sequence with $\|g_t\|_2 \le G$, and any comparator path $u_1, \dots, u_{T'} \in \mathcal{K}$, put $P_2 = \sum_{t=2}^{T'} \|u_t - u_{t-1}\|_2$. Algorithm 1 satisfies, simultaneously for all such paths,*

$$\sum_{t=1}^{T'} \langle g_t, r_t - u_t \rangle \le \frac{3G}{2} \sqrt{H(D^2 + 2DP_2)} + \sqrt{2}\, GD \sqrt{H \log J}. \tag{10}$$

The feedback may depend on the current proposal and all preceding observations.

Proof. For each component k, nonexpansiveness of Euclidean projection and $u_t \in \mathcal{K}$ give

$$\langle g_t, x_{t,k} - u_t \rangle \le \frac{\|x_{t,k} - u_t\|_2^2 - \|x_{t+1,k} - u_t\|_2^2}{2\eta_k} + \frac{\eta_k G^2}{2}.$$

For $x, u, v \in \mathcal{K}$, $|\|x-u\|_2^2 - \|x-v\|_2^2| \le 2D\|u-v\|_2$. The squared-distance terms telescope as

$$\begin{aligned}
&\sum_{t=1}^{T'} \big(\|x_{t,k} - u_t\|_2^2 - \|x_{t+1,k} - u_t\|_2^2 \big) \\
&= \|x_{1,k} - u_1\|_2^2 - \|x_{T'+1,k} - u_{T'}\|_2^2 \\
&\quad + \sum_{t=2}^{T'} \big(\|x_{t,k} - u_t\|_2^2 - \|x_{t,k} - u_{t-1}\|_2^2 \big).
\end{aligned}$$

Bound the initial term by D^2, discard the nonpositive final term, and apply the preceding inequality to each comparator change. Using $T' \le H$ therefore yields

$$\sum_{t=1}^{T'} \langle g_t, x_{t,k} - u_t \rangle \le \frac{D^2 + 2DP_2}{2\eta_k} + \frac{\eta_k G^2 H}{2}.$$

The continuous minimizing rate is $\eta_* = \sqrt{D^2 + 2DP_2}/(G\sqrt{H})$. Since $0 \le P_2 \le D(H-1)$, $\eta_0 \le \eta_* \le \eta_0 \sqrt{2H-1} \le \eta_{J-1}$. The grid is dyadic, so some rate satisfies $\eta_*/2 \le \eta_k \le \eta_*$, giving a component bound of $\frac{3G}{2}\sqrt{H(D^2 + 2DP_2)}$.

For the mixing step, center its losses at x_0: $\widetilde{a}_{t,k} = \langle g_t, x_{t,k} - x_0 \rangle \in [-GD, GD]$. Centering subtracts the same scalar from all component losses, so it does not change the normalized update. The interval has width $2GD$, and Hoeffding's finite-range inequality gives

$$\log \sum_k \pi_{t,k} e^{-\gamma \widetilde{a}_{t,k}} \le -\gamma \sum_k \pi_{t,k} \widetilde{a}_{t,k} + \gamma^2 G^2 D^2 / 2.$$

Let $Q_{t+1} = J^{-1} \sum_k \exp(-\gamma \sum_{s \le t} \widetilde{a}_{s,k})$. The update implies $Q_{t+1}/Q_t = \sum_k \pi_{t,k} e^{-\gamma \widetilde{a}_{t,k}}$, with $Q_1 = 1$. Summing the preceding log-ratio inequality bounds $\log Q_{T'+1}$ above by $-\gamma \sum_{t,k} \pi_{t,k} \widetilde{a}_{t,k} + \gamma^2 G^2 D^2 H/2$. Any component k gives the lower bound $\log Q_{T'+1} \ge -\log J - \gamma \sum_t \widetilde{a}_{t,k}$. Consequently,

$$\sum_{t=1}^{T'} \langle g_t, r_t - x_{t,k} \rangle \le \frac{\log J}{\gamma} + \frac{\gamma G^2 D^2 H}{2} = \sqrt{2}\, GD \sqrt{H \log J}.$$

For the selected k, split $r_t - u_t = (r_t - x_{t,k}) + (x_{t,k} - u_t)$ and add the two bounds. All inequalities hold for each realized feedback sequence and every path, proving simultaneity and permitting adaptive feedback. □

Forest guarantee. For the simplex instantiation in Algorithm 1, the same available-slot adapter gives, simultaneously for every $u_t \in \Delta(a_t)$,

$$R_T(u) \le 4\sqrt{NT[1 + P_T(u) + \log(4 + \log T)]}. \tag{11}$$

Proof. Fix any comparator sequence $u_t \in \Delta(a_t)$ and write $P_2 = \sum_{t=2}^{T} \|u_t - u_{t-1}\|_2$. Since $r_t \in \Delta_N$, we have $z_t = r_t$ and $b_t - \langle u_t, \boldsymbol{l}_t \rangle = \langle g_t, r_t - u_t \rangle$. This identity and Lemma D.2, with $H = T$, $D = \sqrt{2}$, and $G = \sqrt{N}$, give

$$R_T(u) = \sum_{t=1}^{T} [\bar{\ell}_t(\mathbf{w}_t) - \bar{\ell}_t(u_t)] \le 3\sqrt{NT(P_2/\sqrt{2} + 1/2)} + 2\sqrt{NT \log J}.$$

As shown above, $P_2 \le \sqrt{2}\, P_T(u)$. For $T \ge 1$,

$$J \le 2 + \log_2 \sqrt{4 + 8T} \le 2 + \frac{\log 12 + \log T}{2 \log 2} \le 4 + \log T.$$

Writing $p = P_T(u)$, so that $P_2/\sqrt{2} \le p$, Cauchy–Schwarz gives

$$\begin{aligned} 3\sqrt{p + 1/2} + 2\sqrt{\log J} &\le \sqrt{13(p + 1/2 + \log J)} \\ &\le 4\sqrt{p + 1 + \log(4 + \log T)}, \end{aligned}$$

which proves (11) and the ADER part of Theorem 4.1. All inequalities hold for every realized feedback sequence and every comparator sequence, so the comparator may be chosen in hindsight; no probabilistic assumption or union bound is needed. For convex prediction loss, $\ell_t(\mathbf{w}_t) \le \bar{\ell}_t(\mathbf{w}_t)$, which gives the same bound with prediction loss on the learner's side. The comparator is still scored by its weighted individual losses $\bar{\ell}_t(u_t)$; this guarantee makes no claim against the prediction loss of the comparator mixture itself. □

Comparison of the guarantees. Both exact methods compete with the same available-slot comparator paths without knowing $P_T(u)$ in advance. In (11), ADER's $\log(4 + \log T)$ term is additive inside the square root; MDW's bound in (1) multiplies $1 + P_T(u)$ by $\log(eT)$. Thus, for fixed N, ADER's bound is $o(T)$ whenever $P_T(u) = o(T)$, whereas the displayed MDW bound is $o(T)$ when $P_T(u) = o(T/\log(eT))$. These are sufficient conditions from the bounds, not necessary conditions for the algorithms' actual regret.

The reported ADER variant ablation uses four times the reference rates and a 10^{-15} mixing-weight floor, specified in Appendix G; it is distinct from the ideal update proved here. Likewise, the numerical thresholds and exponent cap of the evaluated MDW update are outside its exact-recurrence guarantee. The empirical comparison in Table 13 concerns these evaluated implementations.

E Episode comparison: proof and lifecycle conditions

Proof of Theorem 6.3. We first compare the benchmark with the restarted reference, then add the cost of learning its weights. Fix the reference partition independently of future observations. For $t \in I_s$, put $Z_t = \bar{\ell}_t(u_t) - l_t(m_s^\star)$, and let $V_t \in \{0, 1\}$ mark useful rounds. Both the comparator weights u_t and V_t are $\mathcal{F}_{t^-}$-measurable. Since u_t is a probability vector supported on available models, its weighted individual loss belongs to $[0, 1]$, so $Z_t \in [-1, 1]$. On useful rounds we apply the conditional-quality assumption; on the remaining rounds we only use $Z_t \le 1$. Thus

$$\mathbb{E}[Z_t \mid \mathcal{F}_{t^-}] \le V_t \varepsilon_s + (1 - V_t).$$

On the simultaneous coverage event, each episode has at most g_s rounds with $V_t = 0$. Since $0 \le \varepsilon_s \le 1$,

$$\sum_{t=1}^{T} \mathbb{E}[Z_t \mid \mathcal{F}_{t^-}] \le \sum_{s=1}^{S} \big(g_s + (L_s - g_s)\varepsilon_s\big).$$

Put $M_t = Z_t - \mathbb{E}[Z_t \mid \mathcal{F}_{t^-}]$. Centering preserves the conditional range width, which is at most two, so conditional Hoeffding gives $\mathbb{E}[e^{\lambda M_t} \mid \mathcal{F}_{t^-}] \le e^{\lambda^2/2}$. The nested information sets allow us to iterate this bound across rounds:

$$\mathbb{E}\Big[e^{\lambda \sum_{t=1}^{T} M_t}\Big] \le e^{T\lambda^2/2}.$$

Exponential Markov with $\lambda = \sqrt{2 \log(1/\delta_{\text{loss}})/T}$ then implies, with probability at least $1 - \delta_{\text{loss}}$,

$$\sum_{t=1}^{T} \big(Z_t - \mathbb{E}[Z_t \mid \mathcal{F}_{t^-}]\big) \le \sqrt{2T \log(1/\delta_{\text{loss}})}.$$

For MDW, Theorem 4.1 and convexity give

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \sum_{t=1}^{T} \bar{\ell}_t(u_t) \le c_{\mathrm{DR}} \sqrt{NT[1 + P_T(u)] \log(eT)}.$$

For ADER, the same step uses $4\sqrt{NT[1 + P_T(u) + \log(4 + \log T)]}$ instead. Each bound holds simultaneously over comparator sequences, so the predictable choice of u_t and its random total reallocated weight $P_T(u)$ require no further union bound. Finally,

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - L_T^\star = \sum_{t=1}^{T} [\ell_t(\mathbf{w}_t) - \bar{\ell}_t(u_t)] + \sum_{t=1}^{T} Z_t.$$

Adding the preceding bounds and taking a union bound with the coverage event gives probability at least $1 - \delta_{\mathrm{cov}} - \delta_{\mathrm{loss}}$, with no independence assumption. □

Specialization to a designated challenger. Proposition 6.1 may supply a fresh incumbent, which requires no promotion test but still needs availability and the conditional-quality assumption for useful service. The following specialization instead follows a designated challenger. Choosing the first challenger born at or after an episode boundary permits the birth and paired-evaluation bounds of Proposition A.1. Let $B_s, C_s, a_s \ge 0$ and $\delta_s \in (0, 1)$ be deterministic, with $\sum_s \delta_s \le \delta_{\mathrm{cov}}$. Here B_s bounds the wait for a designated birth, C_s the time allowed for evidence collection, and a_s any additional availability delay. For each episode, designate at birth a particular challenger c_s and its fixed paired incumbent using the information then available. Suppose $\tau_s \le b_s \le \tau_s + B_s$, and fix its test level and power parameters at designation. For the continuous reference test, suppose the conditional hypotheses of Theorem C.1 hold, and that predictable paired evaluation supplies its required observation budget by $b_s + C_s$, unless certification occurs earlier. The pair remains active through this paired evaluation.

Assume a certified candidate can start available service by the next prediction plus at most a_s additional rounds and remains available through the episode. Use a persistent available fallback until then and define the comparator to follow c_s thereafter: set $u_t = e_{i_t}$, where i_t is the slot of the selected model. This selection is predictable because a crossing changes the comparator only at a subsequent prediction. On the candidate-service rounds, assume the theorem's conditional-quality inequality relative to $m_s^\star$. This is the learning assumption; incumbent-relative testing advantage is used separately for certification.

Put $D_s = C_s + a_s$. Conditional power, expectation over the designation history, and a union bound imply that, with probability at least $1 - \sum_s \delta_s$, useful service begins by $\tau_s + B_s + D_s + 1$. Thus

$$g_s = \min\{L_s, B_s + D_s + 1\}$$

is sufficient. If $g_s = L_s$, the fallback alone gives the stated coverage bound, without requiring within-episode certification. If service or persistence has additional failure probability, include it in δ_{cov}; conditional-quality assumptions are imposed on the predictably selected rounds, not after conditioning on eventual success. For this single-model comparator, $P_T(u)$ equals the number of selected-slot changes. With one fallback and one candidate block per episode, and each physical model retaining its slot during service, there are at most $2S$ consecutive blocks and hence $P_T(u) \le 2S - 1$. On the same event, the MDW oracle bound becomes

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - L_T^\star \le \sum_s \left[g_s + (L_s - g_s)\varepsilon_s \right] + c_{\mathrm{DR}} \sqrt{2NST \log(eT)} + \sqrt{2T \log(1/\delta_{\mathrm{loss}})}.$$

The success probability is at least $1 - \delta_{\mathrm{cov}} - \delta_{\mathrm{loss}}$.

Finite lifetimes and handovers. More generally, let $J_{s,1}, \ldots, J_{s,H_s}$ be the consecutive useful service intervals of a predictable single-model comparator $u_t = e_{i_t}$ in episode I_s. Each interval uses a fixed physical model; returning after a gap starts a new interval. The uncovered count is $U_s = L_s - |\bigcup_h J_{s,h}|$. Any justified deterministic envelope $g_s \ge U_s$ on the coverage event can be used in the theorem. Waiting, inter-service gaps, and final gaps after expiry all contribute. If one

persistent fallback serves the gaps, there are at most $2H_s + 1$ physical blocks in episode s. If each service model retains its coordinate, the total reallocated weight is $P_T(u) \le 2\sum_s H_s + S - 1$. The dynamic-regret contribution is therefore at most

$$c_{\mathrm{DR}}\sqrt{NT\left[2\sum_s H_s + S\right]\log(eT)}.$$

Use this term together with the same coverage, approximation, and loss-fluctuation terms in Theorem 6.3.

Joint validity and performance. When the total testing budget is at most α, intersecting the false-certification, coverage, and loss events gives joint validity and the oracle inequality with probability at least $1 - \alpha - \delta_{\mathrm{cov}} - \delta_{\mathrm{loss}}$. The loss inequality itself follows from the comparator, quality, and coverage conditions without requiring the absence of false certifications.

F Convex correction heads with clipped feedback

At a fixed round, write p_u and f_u for the classification prediction and unguarded regression candidate of Section 8, evaluated at coefficient vector u. The correction is learned on $\mathcal{K} = [-\rho, \rho]^d$, with diameter $D = 2\rho\sqrt{d}$. The convex objectives are $\mathcal{L}_t(u) = -\log p_u(y_t)$ for classification and $\mathcal{L}_t(u) = \frac{1}{2}(f_u - \widetilde{y}_t)^2$ for the unguarded regression candidate. Classification uses unfloored log loss in this bound. The ADER construction in Appendix D.3 applies, with the box replacing the simplex. The pilot chooses $G > 0$ before active head rounds, and subsequent gradients are clipped to that norm.

Native predictor at zero correction. The classification head of Section 8 adjusts the native probabilities p_t^{nat} in log-odds space, where $c(i)$ is the class of tree i:

$$p_t^{\mathrm{corr}}(c) = \frac{p_t^{\mathrm{nat}}(c)\exp\left(\sum_{i:c(i)=c} u_{t,i}\phi_{t,i}\right)}{\sum_a p_t^{\mathrm{nat}}(a)\exp\left(\sum_{i:c(i)=a} u_{t,i}\phi_{t,i}\right)}. \tag{12}$$

The native binary probabilities are $\mathrm{softmax}(r, 0) = \mathrm{softmax}(0, -r)$. For $C > 2$, native one-vs-rest votes are $\sigma(-r_c)$, normalized across classes. Consequently the anchors of Algorithm 7 satisfy $p^{\mathrm{nat}} = \mathrm{softmax}(a)$, and

$$\mathrm{softmax}\left(a_c + \sum_{i:c(i)=c} u_i\phi_i\right) = p_u, \qquad p_0 = p^{\mathrm{nat}}, \qquad f_0 = b.$$

The first equality follows by cancelling $\sum_c e^{a_c}$ in (12). Zero lies in the coefficient box. These identities hold at every fixed tree state; holding $u_t = 0$ throughout also recovers the native forecast sequence, since correction updates never change native training. A well-defined regression guard preserves the identity: its clipped correction is zero when $f_0 - b = 0$. The explicit zero-scale completion in Appendix H preserves it as well.

With the cached native advice fixed, the class scores are affine in u. Classification log loss is their log-sum-exp minus the true-class score, while regression half-squared loss is a convex quadratic in u. Differentiating gives the gradients in Algorithm 8.

The next bound applies Lemma D.2 through the standard convexity and diameter argument for clipped feedback.

Proposition F.1 (ADER with clipped gradients)**.** *Let the optimizer horizon $H \ge 1$ and dimension $d \ge 1$ be integers, and let $G, \rho > 0$. Set $\mathcal{K} = [-\rho, \rho]^d$ and $D = 2\rho\sqrt{d}$. Use ADER as in Lemma D.2, with $J = 1 + \lceil \log_2 \sqrt{4 + 8H} \rceil$. On $1 \le T' \le H$ active rounds let $u_t \in \mathcal{K}$ be its decision, let $\mathcal{L}_t$ be convex, and let $g_t \in \partial\mathcal{L}_t(u_t)$ be finite. Feed $\bar{g}_t = g_t \min\{1, G/\|g_t\|_2\}$, with $\bar{g}_t = 0$ when $g_t = 0$.*

For every path $v_t \in \mathcal{K}$, put $P_2 = \sum_{t=2}^{T'} \|v_t - v_{t-1}\|_2$. Then, pathwise,

$$\sum_{t=1}^{T'} [\mathcal{L}_t(u_t) - \mathcal{L}_t(v_t)] \le \frac{3G}{2} \sqrt{H(D^2 + 2DP_2)} + \sqrt{2}\, GD\sqrt{H \log J} + D \sum_{t=1}^{T'} (\|g_t\|_2 - G)_+. \tag{13}$$

Proof. The clipped feedback satisfies $\|\bar{g}_t\|_2 \le G$. The domain, initialization, rate grid, and pre-update meta losses in Algorithms 1 and 8 satisfy Lemma D.2; hence its linear-loss bound applies over the $T' \le H$ active rounds. Convexity and the domain diameter give, at each round,

$$\mathcal{L}_t(u_t) - \mathcal{L}_t(v_t) \le \langle g_t, u_t - v_t \rangle \le \langle \bar{g}_t, u_t - v_t \rangle + D\|g_t - \bar{g}_t\|_2.$$

Radial clipping gives $\|g_t - \bar{g}_t\|_2 = (\|g_t\|_2 - G)_+$. Sum these inequalities and substitute the explicit bound of Lemma D.2. No independence or fixed-in-advance sequence of losses is required for this pathwise argument. □

Pilot envelopes and comparison with the native predictor. For binary classification, $\|g_t\|_2 = |p_{t,1} - \mathbf{1}\{y_t = 1\}|\, \|\phi_t\|_2 \le \|\phi_t\|_2$. For disjoint multiclass feature blocks,

$$\|g_t\|_2^2 = \sum_c (p_{t,c} - \mathbf{1}\{y_t = c\})^2 \|\phi_{t,c}\|_2^2 \le 2 \max_c \|\phi_{t,c}\|_2^2.$$

For regression, the candidate gradient throughout the box obeys

$$\|(b_t + \langle u, \phi_t \rangle - \widetilde{y}_t)\phi_t\|_2 \le (|b_t - \widetilde{y}_t| + \rho\|\phi_t\|_1)\|\phi_t\|_2.$$

Algorithm 8 sets G from these envelopes on the pilot rows and then holds it fixed. Later gradients may exceed G; the final term in (13) accounts for their excess and is zero if no clipping is needed.

Taking the native comparator $v_t = 0$, so $P_2 = 0$, and substituting $D = 2\rho\sqrt{d}$ yields

$$\sum_{t=1}^{T'} [\mathcal{L}_t(u_t) - \mathcal{L}_t(0)] \le \rho G \sqrt{dH}\big(3 + 2\sqrt{2 \log J}\big) + 2\rho\sqrt{d} \sum_{t=1}^{T'} (\|g_t\|_2 - G)_+. \tag{14}$$

Warm-up and pilot rounds add zero excess because they expose native forecasts. The regression statement concerns the unguarded candidate and half squared loss. The additional guard changes the exposed forecast, and trimmed MAE performance is evaluated empirically.

G Forest algorithms and implementation specification

The forest construction has three interfaces: a tree supplies advice, a weighter combines the incumbent advice, and a lifecycle decides which physical trees occupy the slots. The main text analyzes ideal MDW and ADER (Appendix D). The primary ARF-W and ORF runs use the numerical MDW version; Algorithms 2–6 specify that implementation and its shared interfaces. Algorithm 1 gives the ADER alternative in Appendix D.3. Replacing the core changes weighting, while the base-tree and lifecycle rules remain as stated below.

G.1 Inputs, advice, and initialization

Inputs are the task, fixed class order or frozen regression transforms, scored horizon H, and run seed. Each run starts with an untrained forest and freshly initialized optimizer. Regression uses the 500-row calibration transforms of Appendix I.1; subsequent losses and training targets use $\widetilde{y}$, and evaluation forecasts are transformed back to the original units. No regression forecast is clipped before evaluation. The calibration prefix only fits the frozen transforms: it is neither scored nor used for tree training, optimizer updates, or lifecycle advancement. Trees are instantiated before the first scored prediction. The horizon sizes the optimizer grids; reaching it does not trigger a restart.

Base learners. ARF-W, ORF, and the matched ARF controls use River 0.22.0 ARF Hoeffding trees. Table 3 gives their common settings. A tree receives training weight $k \sim \mathrm{Poisson}(6)$ and trains only if $k > 0$. ARF-W consumes one forest random-number generator in tree order. ORF assigns each physical tree a separate generator using the implementation's eight-byte BLAKE2b seed derived from the identifier `seed:slot:role:serial`, where role is incumbent or challenger and serial counts creations for that slot and role. The same generator supplies its leaf feature subsets and Poisson draws; promotion preserves that generator.

Table 3: Base-tree settings shared by the forest methods. F is the number of feature keys presented at the first prediction, including regression one-hot columns. Other behavior follows the named River 0.22.0 base learner.

Setting	Classification	Regression
Feature subset / grace period	$\mathrm{round}(\sqrt{F})$ / 50	same
Depth / binary splits	unlimited / false	same
Split confidence / tie threshold	0.01 / 0.05	same
Split criterion	information gain	variance reduction
Leaf predictor	naive-Bayes adaptive; threshold 0	adaptive mean/linear; decay 0.95
Numeric observer	Gaussian splitter	TEBST-equivalent splitter
Minimum split samples	library default	5
Minimum branch / maximum share	0.01 / 0.99	not applicable
Memory budget (MB)	100	500
Memory-estimate period	2,000,000	same
Merit prepruning / poor-attribute removal	true / false	same
Stop memory management	false	false

Advice and losses. For classification, align each sparse probability mapping to the declared class order, filling absent classes with zero; an empty mapping gives the uniform distribution. Validate its sum and entries to tolerance 10^{-12} and clip only excursions within that tolerance, without renormalizing. Class decisions use the first declared class attaining the maximum. For regression, an untrained tree's `None` prediction is zero. Invalid or non-finite advice aborts a run. For cached advice h and target y, define

	classification	regression
$L(h, y)$	$\frac{1}{2}\|h - e_y\|_2^2$	$\min\{(h - \widetilde{y})^2/4, 1\}$
$S(h, y)$	$\mathbf{1}\{\arg\max h = y\}$	$1 - L(h, y)$

The weighter and paired comparisons use L; the ORF availability sequence uses S. Regression evaluation uses the exposed forecast in original units.

Round and counter conventions. A pending prediction caches the row, all advice, availability, weights, and forecast. Repeated prediction of that row returns the cache without advancing time. Learning must use the same row and consumes the cache exactly once. Slot indices below are $0, \ldots, N-1$. We let q_i count comparisons already opened, increment it before allocating a new level, and never reset it. Paired observations are indexed from zero. These conventions distinguish storage indices from physical birth and deadline times.

G.2 Weighting interface and evaluated optimizer

Algorithm 2: Available-slot prediction and feedback

State: an online core with N coordinates; cache empty. The evaluated version uses Algorithm 3 and $\epsilon_{\mathrm{num}} = 10^{-15}$. The ADER alternative uses Algorithm 1 on Δ_N, $D = \sqrt{2}$, $G = \sqrt{N}$, horizon H, and $\epsilon_{\mathrm{num}} = 0$.

1. **Predict**(a)**:** require $a \in \{0,1\}^N$ and $\sum_i a_i > 0$. Read the core proposal r; set $z = (r)_+$ and $s = \langle a, z\rangle$. If $s > \epsilon_{\mathrm{num}}$, play $w = (a \odot z)/s$; otherwise play $w = a/\sum_i a_i$. Cache (a, w, r, z) and return w.

2. **Update**$(\boldsymbol{l}, a, w)$**:** require a pending prediction, the cached a, w, and $\boldsymbol{l} \in [0,1]^N$. Form $b = \langle w, \boldsymbol{l} \rangle$ and $g = a \odot (\boldsymbol{l} - b\mathbf{1})$. This b is the weighted average of individual losses.
3. Set $d = z - r$ and $h = g$. If $\|d\|_2 > \epsilon_{\text{num}}$ and $\langle g, d \rangle > 0$, set $h = g - \langle g, d \rangle d / \|d\|_2^2$. Feed h to the core, check finite state, and clear the cache.

For the MDW bound in the main theorem, use the ideal core in Appendix D.1 and $\epsilon_{\text{num}} = 0$ in this adapter. For the ADER alternative, the simplex proposal has $d = 0$, so step 3 feeds $h = g$. Every physical-tree replacement leaves this entire weighter state intact.

Algorithm 3: MDW core used in the reported forest runs

Initialize: $G = \sqrt{N}$, $\epsilon = 1$, $V = 4G^2$, and $v_k = 0$ for

$$J_{\text{DR}} = \max\{1, \lceil \log_2 \sqrt{\max(2, H)} \rceil\}, \qquad \eta_k = \min\{2^k/(G\sqrt{H}), 1/G\}, \quad 1 \le k \le J_{\text{DR}}.$$

With $\epsilon_{\text{num}} = 10^{-15}$, define

$$\alpha_{\text{DR}}(V) = \frac{(\epsilon/J_{\text{DR}})G^2}{V\,[\max\{\log \max(V/G^2, 1 + \epsilon_{\text{num}}), \epsilon_{\text{num}}\}]^2}.$$

Predict: return $r = \sum_k v_k$. **Update**(h)**:**

1. Compute $V' = V + \|h\|_2^2$, $\alpha = \alpha_{\text{DR}}(V)$, and $\alpha' = \alpha_{\text{DR}}(V')$.
2. For every k, using the old v_k, put

$$\theta_k = \begin{cases} -h, & \|v_k\|_2 \le \epsilon_{\text{num}}, \\ \dfrac{2\log(1 + \|v_k\|_2/\alpha)}{\eta_k \|v_k\|_2} v_k - h, & \text{otherwise.} \end{cases}$$

   If $\|\theta_k\|_2 \le \epsilon_{\text{num}}$, set $v_k' = 0$. Otherwise set

$$e_k = \min\{\tfrac{1}{2}\eta_k(\|\theta_k\|_2 - 2\eta_k\|h\|_2^2)_+, 700\}, \qquad v_k' = \alpha'(e^{e_k} - 1)\frac{\theta_k}{\|\theta_k\|_2}.$$

3. Store $(V, v_k) \leftarrow (V', v_k')$; abort on any non-finite state.

This is the numerical form of (6): the inverse gradient uses the old scale, and the new point uses the updated scale. Its numerical thresholds and exponent cap are not covered by the real-arithmetic guarantee.

The reported ADER variant. It uses the same expert-loss adapter and simplex component learners as Algorithm 1, with component step sizes multiplied by four. With J components, initial points are $x_{k,1} = \mathbf{1}/N$, initial meta weights are $\pi_{k,1} = 1/J$, and the round counter starts at zero. Before the meta update it replaces each current meta weight π_k by $\max\{\pi_k, 10^{-15}\}$:

$$\pi_k' \propto \max\{\pi_k, 10^{-15}\} \exp(-\gamma \langle g, x_k^{\text{old}} \rangle).$$

The meta rate is unchanged, and the predicted mixture is projected onto the simplex before masking. This identifies the evaluated update variant separately from the exact ADER instantiation in Appendix D.3.

G.3 ARF-W and the ARF controls

Algorithm 4: One cached round of ARF-W

State: $N = 100$ incumbent trees T_i; one shared forest generator; for each i, a warning ADWIN(0.01), drift ADWIN(0.001), optional background tree B_i, metric, and regression residual-variance statistic. Detectors have clock 32, five buckets, minimum window 5, and grace period 10. Initially, all incumbents are untrained, detector and metric states are fresh, and no background trees are present.

1. **Predict**(x)**:** if this row is pending, return its cache without recomputation. Otherwise obtain and cache all incumbent advice h_i. Use Algorithm 2 with $a = \mathbf{1}$ and return $\widehat{y} = \sum_i w_i h_i$ (probability vector in classification).
2. **Learn**(x, y)**:** verify the pending row; calculate $l_i = L(h_i, y)$ from cached advice and update the weighter.

3. For each i in slot order, execute steps 3–5: obtain the tree's native point prediction $\widehat{y}_i = T_i.\texttt{predict_one}(x)$ before training it, and update its Accuracy/MSE bookkeeping metric. Draw $k \sim \text{Poisson}(6)$ from the shared generator. If $k = 0$, continue to the next tree.
4. Train B_i, if present, and then T_i with weight k. For classification set $u = \mathbf{1}\{\widehat{y}_i \neq y\}$. For regression add $e = \widetilde{y} - \widehat{y}_i$ to the native sample-variance statistic; if this is its first residual or its standard deviation is zero, set $u = 1/2$, otherwise set $u = (e + 3s)/(6s)$. This native detector input is not clipped.
5. Update the warning detector with u. On a warning, replace B_i by a fresh tree and reset that detector. Then update the drift detector with u. On drift, replace T_i by B_i if present, or by a fresh tree otherwise; clear the background tree and reset the two detectors and metric. The weighter coordinate is retained.
6. After all slots, clear the row cache.

The native point prediction in step 3 is used for ARF detector/bookkeeping behavior; the cached probability advice in step 2 supplies the weighter loss. In particular, a zero Poisson draw skips both detectors.

ARF100-U uses the same native lifecycle with uniform aggregation instead of the weighter. ARF100-N uses River's native Accuracy-weighted classification vote or regression median; ARF10 makes that substitution with ten trees. These controls retain the original River TEBST object in regression, whereas the forest implementation uses the arithmetic-equivalent optimized splitter.

G.4 ORF pool, score sequences, and paired evidence

The ORF pool has four permanent slots 0–3 and 96 dynamic slots. For powers $p = 0, \ldots, 47$, put $L_p = \max\{50, 2^p\}$ and allocate two slots for $p \leq 5$, three for $6 \leq p \leq 17$, two for $18 \leq p \leq 35$, and one for $p \geq 36$, in increasing-power order. The permanent slots never expire and are always available. Each dynamic slot stores its base lifetime L, physical birth b, cumulative lifetime H_i, deadline e, extension count r, score sequence, metric, persistent q_i, and at most one live challenger. The current incumbent is the only tree in that slot used for prediction. Here H_i is the incumbent's cumulative lifetime, distinct from the optimizer horizon H above.

Fresh installation. $\text{Fresh}(i, c)$ creates a new incumbent tree, its generator, metric, and empty score sequence; sets $(b, H_i, e, r) = (c, L, c + L, 0)$; and clears all challenger state. It preserves q_i and the weighter. Initial installations use $c = 0$ and $q_i = 0$, so $(b, H_i, e, r, q_i) = (0, L, L, 0, 0)$. All 100 initial incumbents are untrained, no challengers are present, and only the four permanent slots are available until dynamic incumbents accumulate qualifying scores. The first scored prediction advances the lifecycle clock from zero to $t = 1$; an initial $L = 50$ slot therefore creates its first challenger at round 25 and has initial deadline 50. A promoted tree retains its physical birth and its learned state rather than calling Fresh.

Score sequence and current availability. Each physical tree has its own availability state, initialized with observation count $n = 0$, $m_0 = 1/2$, $\lambda_1 = 1/2$, and $A_0 = B_0 = Z_0 = R_0 = V_0 = 0$. On its nth score $z_n = S(h, y)$, with $n \geq 1$, perform

$$\psi_n = -\tfrac{1}{4}\{\log(1 - \lambda_n) + \lambda_n\}, \qquad A_n = A_{n-1} + \lambda_n,$$
$$Z_n = Z_{n-1} + z_n, \qquad B_n = B_{n-1} + z_n\lambda_n,$$
$$R_n = R_{n-1} + 4(z_n - m_{n-1})^2\psi_n, \qquad m_n = \frac{1/2 + Z_n}{n+1},$$
$$V_n = V_{n-1} + (z_n - m_n)^2, \qquad \sigma_n^2 = \frac{1/4 + V_n}{n+1},$$
$$\lambda_{n+1} = \min\left\{\sqrt{\frac{2\log 10}{\sigma_n^2(n+1)\log(n+2)}}, \frac{1}{2}\right\}, \qquad \text{LCB}_n = \frac{B_n - \log 10 - R_n}{A_n}.$$

These are the documented PMEB recursions with nominal level 0.1 [Waudby-Smith and Ramdas, 2022]. We use LCB_n only as a pre-outcome availability statistic; the dynamic-regret guarantee requires no current-risk or future-performance guarantee for it. At prediction time a dynamic incumbent is available iff $n > 0$ and its *current* $\text{LCB}_n \geq 0$. There is no permanent admission flag. Retention keeps the sequence and promotion transfers the challenger's sequence. Four permanent slots ensure a nonempty available pool.

Paired four-bet subroutine. In this subroutine, α is the test level and D the number of paired observations available before the deadline, rather than a domain diameter. At a challenger birth, increment q_i, assign $\alpha = 0.05/[96q_i(q_i+1)]$, set the comparison budget $D = e - t$, and initialize four component log wealths $U_\beta = V_\beta = V_\beta^{\text{late}} = 0$ for $\beta \in \{1/8, 1/4, 1/2, 1\}$. Let $j = 0$ denote the next pair index and $j_{\text{late}} = \lfloor D/2 \rfloor$. On each cached loss pair, put $d = l^{\text{inc}} - l^{\text{ch}}$ and update

$$U_\beta \leftarrow U_\beta + \log(1+\beta d), \qquad V_\beta \leftarrow V_\beta + \log(1-\beta d).$$

If $j \geq j_{\text{late}}$, also add $\log(1-\beta d)$ to V_β^{late}; then increment j. Use $\log 0 = -\infty$ and stable log-sum-exp to form

$$\log E = \text{LSE}_\beta(U_\beta) - \log 4, \qquad \log E^{\text{keep}} = \text{LSE}_\beta(V_\beta^{\text{late}}) - \log 4.$$

Certification is $\log E \geq \log(1/\alpha)$; a comparison with no late observation has $\log E^{\text{keep}} = 0$. The forward wealth E has the null-supermartingale property proved in Appendix B.1. The reverse full-window wealth is recorded but does not enter the retention decision with L-round extensions.

Algorithm 5: One cached round of ORF

State: the pool and persistent weighter above, with one generator per physical tree and current lifecycle counter t.

1. **Predict**(x)**:** if this row is pending, return its cached forecast. Otherwise advance t once and execute Algorithm 6. Cache incumbent advice h_i and each live challenger's advice h_i^{ch}. Read the availability mask from the permanent status/current score sequence. Call Algorithm 2 and return $\sum_i w_i h_i$. Challenger advice is used only for its own score and paired comparison.
2. **Learn**(x, y)**:** verify the pending row. Compute every incumbent loss $l_i = L(h_i, y)$ and update the weighter using the cached mask and weights.
3. For each slot i in order, execute steps 3–4: increment the incumbent's bookkeeping count, update its metric and availability state with $S(h_i, y)$, and, if a challenger exists, update that pair with $(l_i, L(h_i^{\text{ch}}, y))$.
4. Draw k from the incumbent's generator and train it with weight k if $k > 0$. If a challenger exists, then update its metric and availability state, draw from its own generator, and train it if its draw is positive.
5. Clear the row cache. Use evidence from this row at the next performed lifecycle pass; do not replace any cached model during this update.

All live trees receive score updates and every active pair receives an observation even when one or both Poisson training draws are zero.

Algorithm 6: ORF lifecycle before prediction at time t

For each dynamic slot in index order, execute the following steps once. Write (b, H_i, e, r) for its incumbent metadata and L for its fixed base lifetime.

1. **Birth.** If no challenger is active and $t \geq b + \lfloor H_i/2 \rfloor$, create one fresh challenger with physical birth t, its own generator, metric and empty availability state, and the paired test above. Mark the comparison active. At most one challenger is active per slot.
2. **Certified promotion.** If the active comparison is certified, replace the incumbent with that challenger, preserving its physical birth, tree state, generator, metric, and availability state. Set $(H_i, e, r) = (L, t+L, 0)$, clear all challenger/test state, and finish this slot's pass.
3. **Deadline.** Otherwise, if $t \geq e$ and a challenger exists, retain the incumbent iff the test has at least one observation, $\log E^{\text{keep}} > 0$, and $r < 64$. On retention set $(H_i, e, r) = (H_i + L, t + L, r + 1)$ and discard the challenger/test. Preserve the incumbent's birth, tree, score sequence, and generator. If retention fails, call $\text{Fresh}(i, t)$ instead. An expired slot without a challenger also calls $\text{Fresh}(i, t)$.
4. Otherwise leave the slot unchanged. Every transition preserves the slot's comparison count and weighting state. Do not rerun the birth check after a promotion, retention, or fresh installation in this pass.

Certification is checked before deadline retention, including evidence from the last comparison observation. A retention therefore starts its next comparison on the next lifecycle pass, without another half-lifetime wait; a promoted tree keeps its original birth clock.

G.5 Controls and ablation substitutions

Table 4: Changes defining the principal forest controls and ablations. Unchanged components follow the algorithms above. The exact ADER reference and the two documented optimizer ablations are identified separately.

Method/arm	Substitution
ARF-W	MDW weights; native ARF tree/detector lifecycle (Algorithm 4); all slots available.
ORF	MDW weights; retention with L-round extensions; Algorithms 5 and 6.
ADER instantiation	Replace the core by Algorithm 1; use its exact available-slot reduction in Appendix D.3.
ARF100-U	Native ARF lifecycle; uniform classification vote or regression mean.
ARF100-N / ARF10	Native ARF lifecycle and aggregation; 100 / 10 trees.
Uniform vote on ORF trees	Replay the same physical trees and availability; use $w_i = a_i / \sum_j a_j$.
Renewal without challengers	Suppress challengers and tests; call $\mathrm{Fresh}(i, t)$ at every dynamic deadline.
No deadline retention	Keep challenger births and certified promotion; call $\mathrm{Fresh}(i, t)$ at each uncrossed deadline.
No certified promotion	Suppress certified promotions; retain the paired comparisons for the ordinary deadline-retention rule.
Random deadline retention	Keep certified promotion; use the sampling rule below, with probabilities estimated from saved ORF decisions per stream and slot power.
ORF (ADER variant)	Use the documented fourfold-rate, floored-meta-weight variant above.
ORF (alternative DR update)	Use Algorithm 3 with the updated scale α' replacing α in the expression for θ_k; all other updates and initialization are unchanged.

Alternative DR update. The alternative DR update uses the same rate grid, initialization, available-slot adapter, and numerical safeguards as MDW. Its sole change is to use $\alpha' = \alpha_{\mathrm{DR}}(V + \|h\|_2^2)$ in both the inverse-gradient calculation for θ_k and the new-radius calculation in Algorithm 3; MDW uses the old scale in the former. The exact MDW guarantee does not cover this altered recurrence.

Random deadline retention. For each stream d and slot power p, let $K_{d,p}$ and $F_{d,p}$ count retentions and deadline resets in the saved ORF event logs. Set $\rho_{d,p} = K_{d,p}/(K_{d,p} + F_{d,p})$ when the denominator is positive; otherwise use the stream-wide ratio $\sum_p K_{d,p} / \sum_p (K_{d,p} + F_{d,p})$. Counts pool seeds 1–3 for 25 streams. For Bank Marketing, Gas Turbine, Hotel Bookings, Household Power, NOAA Weather, Online News, Rialto, and Steel Energy, they use instrumented seed-1 reruns whose predictions reproduced the saved runs exactly. These probabilities are fixed before running the control. Each forest uses a separate Python `random.Random` generator initialized with its model seed, shared across slots in lifecycle order and preserved across tree replacements. At an uncertified deadline with comparison observations, draw $U \sim \mathrm{Uniform}[0, 1)$ and retain iff $U < \rho_{d,p}$ and $r < 64$. The draw precedes the cap check; a deadline without comparison observations renews without a draw. Certified promotion and all other lifecycle rules are unchanged. There is one realization per model seed (1–3). The control matches retention probabilities, not the realized counts, timing, or identities of retained trees.

H SGBT/SGBR and the detached correction: algorithms

The native learner and the correction have separate state. The native learner is the configured MOA SGBT/SGBR model; it receives every labeled row and retains its feature subspaces, bagging, stage updates, and internal tree replacement. The head reads its pre-label predictions and updates only its own coefficients. The response below is the class label or, in regression, the standardized target $\widetilde{y}$; output conversion back to original units occurs after the head and guard. In classification, the anchor a is a score vector satisfying $\mathrm{softmax}(a) = p^{\mathrm{nat}}$, so zero correction preserves the native probabilities.

How boosting and bagging combine. At stage m, OzaBag resamples the incoming training example separately for its B members using Poisson(1) counts. Their mean is the stage prediction, and the M stage predictions add to the committee score [Oza and Russell, 2001, Gunasekara et al., 2025]. Boosting trains later stages on derivative-based residual targets computed from preceding

stages. In conventional additive-score notation, if $g = \partial_z L(y, z)$ and $h = \partial_z^2 L(y, z) > 0$, then

$$gq + \tfrac{1}{2}hq^2 = \tfrac{1}{2}h(q + g/h)^2 - \tfrac{g^2}{2h}.$$

Thus the local quadratic target is $-g/h$, with weight h [Gunasekara et al., 2024]. This explains the Newton-style stage objective; the unchanged native training call implements its own score orientation, learning rate, and Hessian handling. The correction uses the documented *raw* member outputs at inference: no additional learning-rate factor multiplies the sums in Algorithm 7.

Algorithm 7: Extract native advice and correction features

Input: a fixed pre-label native-model state, feature vector x, M stages, B members per stage, and $C \geq 2$ classes when applicable.

1. For each native committee c and stage m, construct exactly the native stage sub-instance (its feature subspace and encoding). Read every member's raw prediction $v_{c,m,j}$, $j = 1, \ldots, B$, and cache

   $$h_{c,m} = B^{-1} \sum_j v_{c,m,j}, \qquad r_c = \sum_m h_{c,m}.$$

   The reported runner checks agreement with native stage and committee scores to tolerance 10^{-12}.
2. **Binary classification:** there is one committee. Set $a = (0, -r)$, $p^{\mathrm{nat}} = \mathrm{softmax}(a)$, $d = MB$, and $\phi_{m,j} = -v_{m,j}/B$; assign every coordinate to class 1. This equals native $\mathrm{softmax}(r, 0)$, with class indices 0 and 1.
3. **Multiclass classification:** use one native committee per class. Set $a_c = \log \sigma(-r_c)$, $p^{\mathrm{nat}} = \mathrm{softmax}(a)$, and $\phi_{c,m,j} = -v_{c,m,j}/B$, assigned to class c, with $d = CMB$. This is the native normalization of positive one-vs-rest votes. Here $\sigma(z) = (1 + \exp(-z))^{-1}$; evaluate $\log \sigma$ stably.
4. **Regression:** there is one committee. Set $b = r$, $d = MB$, and $\phi_{m,j} = v_{m,j}/B$.
5. Return and retain the advice, anchor/baseline, and feature vector for the pending outcome. Classification anchor probabilities are also checked against native probabilities to tolerance 10^{-12}.

Algorithm 8: Prequential detached correction of SGBT/SGBR

Inputs: configured native learner, scored horizon $n \geq 10$, and $(\rho, c_G) = (2, 4)$ for classification or $(0.5, 1)$ for regression. Let $w = \min\{200, \lfloor n/10 \rfloor\}$ and $q = \min\{500, \lfloor n/10 \rfloor\}$. **State:** pilot maximum $B_{\max} = 0$, regression pilot squared-residual sum $R = 0$; deployed ADER and guard are created after the pilot. For scored rounds $t = 1, \ldots, n$:

1. **Activate after the pilot.** At $t = w + q + 1$, set $G = \max\{10^{-6}, c_G B_{\max}\}$ and initialize Algorithm 1 on $\mathcal{K} = [-\rho, \rho]^d$, with $D = 2\rho\sqrt{d}$, horizon $H = n$, and initial point 0. For regression also set $s_0^2 = R/q$, $s^2 = s_0^2$, and $\mathcal{C} = \mathcal{Q} = 0$.
2. **Predict before observing the response.** During warm-up $t \leq w$, expose the native forecast. On subsequent rows, cache Algorithm 7's outputs. During the pilot $w < t \leq w + q$, again expose the native forecast. On active rounds $t > w + q$, obtain ADER's coefficient vector u_t and compute

   $$p_t = \mathrm{softmax}\left(a_c + \sum_{i:c(i)=c} u_{t,i}\phi_i\right) \quad \text{or} \quad f_t = b + \langle u_t, \phi \rangle.$$

   Expose p_t in classification; in regression expose $b + \lambda\delta$ from the causal guard below, caching the unguarded candidate f_t and capped correction δ.
3. **Observe and score.** Receive the outcome and score the previously exposed forecast using the evaluation metrics and units in Appendix I.1.
4. **Collect pilot statistics, if $w < t \leq w + q$.** Update $B_{\max}$ with the maximum of its old value and

   $$B_t = \begin{cases} \|\phi\|_2, & \text{binary classification,} \\ \sqrt{2}\max_c \|\phi_c\|_2, & \text{multiclass classification,} \\ (|b - \widetilde{y}| + \rho\|\phi\|_1)\|\phi\|_2, & \text{regression.} \end{cases}$$

   For regression also set $R \leftarrow R + (b - \widetilde{y})^2$. No deployed coefficient update uses these pilot rows.

5. **Update the active head, if $t > w + q$.** Use the cached pre-label advice to form

$$g_i = \phi_i(p_{t,c(i)} - \mathbf{1}\{y = c(i)\}) \quad \text{or} \quad g = (f_t - \widetilde{y})\phi.$$

Set $\bar{g} = g \min\{1, G/\|g\|_2\}$, with $\bar{g} = 0$ for $g = 0$, count a clipped round when $\|g\|_2 > G$, and pass $\bar{g}$ to Algorithm 1. Its meta losses use the component points *before* their projected updates. Update the regression guard from the cached δ and residual $e = b - \widetilde{y}$.

6. **Train the native learner.** On every round, including warm-up and pilot, call the configured native model's training method once with (x, y) or $(x, \widetilde{y})$. This call computes native stage targets and performs bagging/tree updates. It receives no corrected prediction, head gradient, or guarded residual as a replacement target.

Causal regression guard. Use the previous state $(\mathcal{C}, \mathcal{Q}, s^2)$, with $s = \sqrt{s^2}$, to predict:

$$\delta = \mathrm{clip}(f_t - b, -4s, 4s), \qquad \lambda = \mathrm{clip}\left(-\frac{\mathcal{C}}{\mathcal{Q} + 0.1s^2}, 0, 1\right), \qquad \widehat{y}_t = b + \lambda\delta. \tag{15}$$

For a complete mathematical specification, return b directly when $f_t - b = 0$, and set $\lambda = 0$ when its denominator is zero. These are explicit totalizing conventions. After observing the standardized target, use the old scale in

$$\begin{aligned} &e = b - \widetilde{y}, \qquad \kappa = \exp(-1/500), \quad \kappa' = \exp(-1/2000), \\ &\mathcal{C} \leftarrow \kappa\mathcal{C} + (1-\kappa)e\delta, \qquad \mathcal{Q} \leftarrow \kappa\mathcal{Q} + (1-\kappa)\delta^2, \\ &s^2 \leftarrow \max\{10^{-12}s_0^2,\ \kappa' s^2 + (1-\kappa')\,\mathrm{clip}(e, -3s, 3s)^2\}. \end{aligned}$$

Thus all forecast-time guard values depend only on earlier outcomes. If $s_0 > 0$, the denominator stays positive, so the zero-denominator convention is unused. If $s_0 = 0$, the stated recursion keeps $s = \delta = 0$, so the exposed forecast remains native. The ADER gradient is evaluated at f_t, not at the guarded forecast $\widehat{y}_t$.

Native configuration and driver state. The matched runs use MOA through CapyMOA 0.11, a 75% stage feature fraction, one-hot handling enabled internally, classification squared-loss mode disabled, and learning rate 0.1 for classification or 1 for regression. Each stage is `meta.OzaBag -s B` with FIMT-DD members having grace period 50, confidence 0.01, variance-reduction splits, and the `-e` option. Appendix I.6 gives the committee shapes. Declared nominal inputs use ordinal codes; regression features are unscaled and the target uses the same fixed prefix transform as the forests.

The evaluated implementation also updates a shadow controller in its detached advice driver; its coefficients are never exposed. Algorithm 8 shows the separate deployed controller that determines forecasts. The reported whole-process timing includes the driver's extra work.

I Experimental details, additional results, and limitations

This appendix gives the protocol and supporting results for the 33 real-world datasets. Unless stated otherwise, scores are means over model seeds 1–3. All dataset averages give equal weight to each stream.

I.1 Protocol and configurations

Prediction, scoring, and preprocessing. Each row is predicted before its outcome is used for training, preserving the source row order across methods and seeds. Classification half-Brier is $\frac{1}{2}\sum_c (p_c - y_c)^2 \in [0, 1]$, with one-hot y; accuracy uses the largest predicted probability, and log loss floors the true-class probability at 10^{-15}. Regression MAE is measured in original target units. Tail-robust (trimmed) MAE removes each run's largest 0.1% of absolute errors (at least one row). Trimming does not define an online loss.

For regression, the first 500 unscored rows fix the transforms. The target is centered at its median and scaled by its interquartile range divided by 1.3489795003921634, with population standard deviation and then one as fallbacks. ARF-based methods standardize numerical features by prefix means and population standard deviations, using unit scale for constant features. Declared nominal features are

one-hot encoded in regression and remain native in classification. Boosting uses ordinal nominal codes and unstandardized features, with the same target transform. Internal regression feedback uses standardized units; evaluation forecasts are transformed back without clipping. The booster's scored warm-up and pilot are separate from this calibration prefix. Table 5 gives all scored horizons.

For three-seed mean losses $\bar{L}_d^A, \bar{L}_d^B$, the pooled ratio is $\exp\{|\mathcal{D}|^{-1} \sum_{d\in\mathcal{D}} \log(\bar{L}_d^A/\bar{L}_d^B)\}$. The primary forest and boosting summaries use all 17 classification and 16 regression datasets. Reductions are one minus this ratio. Main-table forest summaries use native ARF100 as reference; ablations state their own denominators, so their percentage reductions are not additive.

Forest configurations. Native ARF100 (ARF100-N) uses running-accuracy-weighted classification probabilities and an unweighted regression median. Uniform ARF100 (ARF100-U) uses arithmetic means with the same tree and detector settings. ARF-W retains ARF's warning/drift detectors and background trees, replacing aggregation with MDW. ORF uses the same weighter with four permanent and 96 dynamic slots and at most one challenger per dynamic slot. Its lifecycle uses the lifetime allocation specified in Appendix G, equal four-fraction betting, persistent levels $0.05/[96q(q+1)]$, and at most 64 deadline retentions.

Both methods use $G = 10$, budget $\epsilon = 1$, and the scored horizon for the optimizer. Feedback is individual-tree half-Brier or capped squared loss $\min\{(f_i - \widetilde{y})^2/4, 1\}$. Slot-weight history survives replacement. Lifecycle decisions precede prediction, evidence uses cached predictions, and training follows scoring. Paired observations count even when a tree's Poisson training draw is zero. Appendix G gives the tree settings, availability rule, numerical safeguards, and complete updates. ARF-W preserves ARF's shared tree random-number generator, whereas ORF uses separate generators per physical tree; their full comparison includes this implementation difference.

Software and resources. Forest methods use River 0.22.0, NumPy 2.1.3, and SciPy 1.15.2 on Python 3.13. Boosting uses CapyMOA 0.11 on Python 3.10 and Java 17; the OSB and SOKNL baselines of Appendix I.3 use CapyMOA 0.11 on Python 3.13 and Java 21. The resource study used an AWS c8a.16xlarge with 64 AMD EPYC 9R45 vCPUs, 123 GiB memory, and no swap; numerical libraries were single-threaded. CPU measurements include all process threads, and peak resident memory includes JVM overhead. The same-load measurements retain differences in runners, encodings, and total trained models.

I.2 Real-world results

Table 1 contains the complete primary forest comparison. The following tables give dataset sources, classification accuracy, and log loss. Missing source values are shown as dashes. Rankings may differ across metrics; the main claims use half-Brier and tail-robust MAE.

Table 5: Dataset catalog and scored horizons for the 33 streams. Counts are reported before encoding; links identify the sources. Regression scored rows exclude 500 calibration rows.

Stream	Source rows	Scored rows	Features	Classes	Link
Classification					
Bank Marketing	45,211	45,211	15	2	source
Covtype	581,012	50,000	54	7	source
EEG Eye State	14,980	14,980	14	2	source
Elec2	45,312	45,312	8	2	source
Gas Sensor Drift	13,910	13,910	128	6	source
HTTP-KDD99	567,498	50,000	3	2	source
Hotel Bookings	119,390	119,390	25	2	source
INSECTS abrupt	52,848	52,848	33	6	source
INSECTS gradual	24,150	24,150	33	6	source
INSECTS recurring	79,986	79,986	33	6	source
Image Segments	2,310	2,310	18	7	source
NOAA Weather	18,159	18,159	8	2	source
Occupancy Detection	20,560	20,560	8	2	source
Phishing	1,250	1,250	9	2	source
Rialto	82,250	82,250	27	10	source
Room Occupancy	10,129	10,129	18	4	source
Spam	9,324	9,324	39,916	2	source
Regression					
Abalone	4,177	3,677	8	—	source
Air Quality CO	7,674	7,174	16	—	source
Appliances Energy	19,735	19,235	21	—	source
Beijing $PM_{2.5}$	41,757	41,257	13	—	source
Bike	17,379	16,879	12	—	source
Demand	150,150	50,000	8	—	source
Gas Turbine	36,733	36,233	9	—	source
House8L	22,784	22,284	8	—	source
Household Power	2,049,280	200,000	8	—	source
Metro	48,204	47,704	7	—	source
NZEnergy	3,100,666	20,000	10	—	source
Online News	39,644	39,144	58	—	source
Seoul Bike	8,760	8,260	13	—	source
Steel Energy	35,040	34,540	9	—	source
Tetouan Power	52,416	51,916	8	—	source
Trump Approval	1,001	501	6	—	source

Table 6: Classification accuracy in percent (higher is better). All scores are means over seeds 1–3.

Stream	ARF100-N	ARF100-U	ARF-W	ORF
Bank Marketing	89.39	89.4	89.42	89.27
Covtype	87.78	87.7	90.91	90.79
EEG Eye State	97.77	97.77	99.03	99.25
Elec2	90	89.96	93.01	93.26
Gas Sensor Drift	81.36	80.89	89.57	95.67
HTTP-KDD99	100	100	100	100
Hotel Bookings	84.28	84.16	88.39	91.58
INSECTS abrupt	76.06	76.02	76.05	76.08
INSECTS gradual	78.67	78.53	79.18	78.56
INSECTS recurring	78.82	78.77	79.05	78.45
Image Segments	82.81	82.54	85.4	89.54
NOAA Weather	79.58	79.57	79.61	79.46
Occupancy Detection	98.76	98.76	99.22	99.27
Phishing	91.33	91.31	90.83	90.64
Rialto	83.47	81.97	81.42	86.82
Room Occupancy	98.53	98.54	99.42	99.52
Spam	88.72	88.5	94.71	95.13

Table 7: Classification log loss, with true-class probability floored at 10^{-15} (lower is better). C: SGBT + correction. All scores are means over seeds 1–3; dashes denote unreported values.

Stream	ARF100-N	ARF100-U	ARF-W	ORF	C 5×20	C 10×10
Bank Marketing	—	—	0.2868	0.2901	0.312	0.3573
Covtype	0.4060	0.4155	0.2845	0.3110	0.2481	0.3132
EEG Eye State	0.0824	0.0829	0.0716	0.0632	0.0351	0.0288
Elec2	0.2826	0.2846	0.2065	0.2201	0.2177	0.2144
Gas Sensor Drift	0.5920	0.6193	0.4006	0.2718	0.3288	0.4174
HTTP-KDD99	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Hotel Bookings	—	—	0.3303	0.3887	0.2628	0.2472
INSECTS abrupt	0.7285	0.7355	0.6802	0.6797	0.7511	0.9441
INSECTS gradual	0.6650	0.6714	0.6056	0.6122	0.6677	0.8464
INSECTS recurring	0.6348	0.6393	0.5936	0.6100	0.6455	0.7944
Image Segments	0.7334	0.7595	0.5297	0.4595	0.4304	0.6188
NOAA Weather	—	—	0.4504	0.4513	0.4744	0.513
Occupancy Detection	0.0413	0.0417	0.0294	0.0255	0.0264	0.0282
Phishing	0.2900	0.2919	0.2715	0.2754	0.3460	0.4834
Rialto	—	—	0.6986	0.5736	0.4606	0.5473
Room Occupancy	0.0720	0.0723	0.0488	0.0311	0.1031	0.1285
Spam	0.2850	0.2891	0.1887	0.1967	0.1121	0.1122

I.3 Additional baselines

Beyond the two ARF100 controls, we ran the streaming ensembles named in the related work that have maintained implementations in River or CapyMOA: Streaming Random Patches (SRP) [Gomes et al., 2019] in both tasks, OzaBoost [Oza and Russell, 2001] and Online Smooth Boosting (OSB) [Chen et al., 2012] in classification, SOKNL [Sun et al., 2022] in regression, and a ten-tree ARF (ARF10) as a size control. SRP and ARF10 use River 0.22 with 100 and 10 members, grace period 50, Poisson rate $\lambda = 6$, the forests' leaf rule (naive-Bayes-adaptive leaves in classification, adaptive leaves in regression), and River's defaults otherwise; OzaBoost is River's `AdaBoostClassifier` over 100 such Hoeffding trees with its own example weights. OSB (MOA `OnlineSmoothBoost`, 100 Hoeffding trees, $\gamma = 0.1$) and SOKNL (100 trees, 60% feature subspaces, ADWIN detectors, self-optimising k) run through CapyMOA 0.11 [Gomes et al., 2025] on Java 21. Every run scored the same rows as Table 1 with seeds 1–3 on all 33 streams and had four CPU-hours. SRP and OzaBoost

did not finish Spam within that budget (fewer than 1,100 of its 9,324 rows; DNF below). OSB returned non-finite votes from MOA at the same row for all three seeds on 14 of the 17 classification streams, as early as the third row; the cause was not traced and no workaround was applied, so these runs count as failed and OSB has values on EEG Eye State, HTTP-KDD99, and Phishing only. SGBT and SGBR are compared in Appendix I.6.

Tables 9–11 give every value, and Table 8 pools the per-stream ratios over the streams each baseline finished. ORF has lower half-Brier than ARF10, both ARF100 controls, OzaBoost, and OSB on every stream they finished, with one tie on HTTP-KDD99; SRP has lower half-Brier than ORF and ARF-W on Bank Marketing and NOAA Weather. In regression, SRP has lower tail-robust MAE than ORF on Air Quality CO, Gas Turbine, and Trump Approval, and ARF10 and native ARF100 on House8L and Online News; SOKNL never does. SRP's pooled regression ratio is dominated by Metro, where its error is three orders of magnitude above the other methods', so its median change is the more representative summary. Error rate ranks differently from half-Brier: native ARF100 has lower error than ORF on Bank Marketing, INSECTS gradual, INSECTS recurring, NOAA Weather, and Phishing, and OzaBoost on EEG Eye State, Occupancy Detection, and Room Occupancy, although OzaBoost's half-Brier is higher on every stream (by a factor of 3.1 in geometric mean), since its votes are poorly calibrated.

Table 8: **Pooled comparison with the additional baselines on all 33 streams.** Geometric means of per-stream ratios of three-seed mean losses, ORF or ARF-W over the baseline, over the streams the baseline finished; negative favors ORF or ARF-W. W/L/T counts streams where ORF (or ARF-W) is better, worse, or tied. HTTP-KDD99 has no error-rate ratio (every method has error zero).

Baseline	Metric	Streams	ORF	W/L/T	Median	ARF-W	W/L/T	Median
Classification								
ARF10	Half-Brier	17	−37.1%	16/0/1	−39.9%	−30.7%	16/0/1	−31.9%
ARF10	Error rate	16	−42.2%	16/0/0	−43.1%	−32.9%	16/0/0	−32.0%
ARF100-N	Half-Brier	17	−32.8%	16/0/1	−34.0%	−25.9%	16/0/1	−24.8%
ARF100-N	Error rate	16	−35.2%	11/5/0	−28.6%	−24.7%	13/3/0	−20.3%
ARF100-U	Half-Brier	17	−34.0%	16/0/1	−34.4%	−27.1%	16/0/1	−30.9%
ARF100-U	Error rate	16	−35.9%	12/4/0	−29.9%	−25.6%	14/2/0	−21.2%
SRP	Half-Brier	16	−35.4%	13/2/1	−19.5%	−28.7%	13/2/1	−18.2%
SRP	Error rate	15	−40.3%	14/1/0	−18.6%	−30.4%	15/0/0	−20.9%
OzaBoost	Half-Brier	16	−68.1%	16/0/0	−60.9%	−64.7%	16/0/0	−58.5%
OzaBoost	Error rate	15	−47.0%	12/3/0	−43.2%	−38.1%	12/3/0	−38.3%
OSB	Half-Brier	3	−75.7%	3/0/0	−66.7%	−73.8%	3/0/0	−66.7%
OSB	Error rate	2	−72.9%	2/0/0	−50.6%	−69.5%	2/0/0	−50.3%
Regression								
ARF10	Tail-robust MAE	16	−30.0%	14/2/0	−24.1%	−25.0%	14/2/0	−21.8%
ARF100-N	Tail-robust MAE	16	−26.3%	14/2/0	−20.8%	−21.1%	14/2/0	−17.8%
ARF100-U	Tail-robust MAE	16	−34.5%	16/0/0	−25.6%	−29.8%	16/0/0	−22.2%
SRP	Tail-robust MAE	16	−47.1%	13/3/0	−11.7%	−43.4%	11/5/0	−6.5%
SOKNL	Tail-robust MAE	16	−49.8%	16/0/0	−41.1%	−46.2%	15/1/0	−40.5%

Table 9: **Classification half-Brier with the additional baselines.** All 17 streams; means over seeds 1–3, lower is better, best in bold. DNF: the four CPU-hour limit was reached. Failed: MOA returned non-finite votes.

Stream	ARF10	ARF100-N	ARF100-U	SRP	OzaBoost	OSB	ARF-W	ORF
Bank Marketing	0.0843	0.08189	0.0819	**0.08108**	0.1828	Failed	0.08139	0.0815
Covtype	0.1103	0.103	0.1053	0.08452	0.1949	Failed	**0.0714**	0.07461
EEG Eye State	0.01779	0.017	0.01719	0.0183	0.04014	0.09328	0.007691	**0.00609**
Elec2	0.08877	0.08083	0.08138	0.07898	0.1353	Failed	0.05504	**0.05335**
Gas Sensor Drift	0.1588	0.141	0.1475	0.2414	0.3456	Failed	0.08749	**0.05221**
Hotel Bookings	0.1265	0.1201	0.121	0.1225	0.1805	Failed	0.09053	**0.06987**
HTTP-KDD99	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}	6.7×10^{-6}	1.5×10^{-5}	5.0×10^{-6}	5.0×10^{-6}
Image Segments	0.1661	0.1561	0.162	0.2001	0.3637	Failed	0.1093	**0.08884**
INSECTS abrupt	0.1939	0.1853	0.1868	0.2015	0.3073	Failed	**0.1759**	0.1766
INSECTS gradual	0.1696	0.1636	0.1651	0.187	0.3083	Failed	**0.1517**	0.1534
INSECTS recurring	0.1666	0.1599	0.161	0.1848	0.3047	Failed	**0.1523**	0.1576
NOAA Weather	0.1514	0.1456	0.1457	**0.1441**	0.2173	Failed	0.1448	0.1452
Occupancy Detection	0.01082	0.009667	0.0097	0.007445	0.05216	Failed	0.006377	**0.005877**
Phishing	0.08231	0.07371	0.07418	0.07338	0.1669	0.1081	**0.07077**	0.07171
Rialto	0.2161	0.1957	0.2131	0.2621	0.4353	Failed	0.1472	**0.1269**
Room Occupancy	0.01379	0.01188	0.01208	0.01058	0.04986	Failed	0.004781	**0.003843**
Spam	0.08715	0.08515	0.08672	DNF	DNF	Failed	0.04686	**0.04305**

Table 10: **Classification accuracy in percent with the additional baselines.** Means over seeds 1–3; higher is better, best in bold.

Stream	ARF10	ARF100-N	ARF100-U	SRP	OzaBoost	OSB	ARF-W	ORF
Bank Marketing	89.02	89.39	89.4	89.4	83.33	Failed	**89.42**	89.27
Covtype	85.78	87.78	87.7	89.27	81.83	Failed	**90.91**	90.79
EEG Eye State	97.75	97.77	97.77	97.66	**99.51**	90.73	99.03	99.25
Elec2	88.58	90	89.96	90.52	92.05	Failed	93.01	**93.26**
Gas Sensor Drift	78.63	81.36	80.89	71.71	60.86	Failed	89.57	**95.67**
Hotel Bookings	82.65	84.28	84.16	83.45	81.17	Failed	88.39	**91.58**
HTTP-KDD99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Image Segments	78.85	82.81	82.54	74.26	29.35	Failed	85.4	**89.54**
INSECTS abrupt	74.28	76.06	76.02	71.87	57.92	Failed	76.05	**76.08**
INSECTS gradual	77.27	78.67	78.53	73.73	56.63	Failed	**79.18**	78.56
INSECTS recurring	77.5	78.82	78.77	73.52	55.06	Failed	**79.05**	78.45
NOAA Weather	78.3	79.58	79.57	79.44	76.02	Failed	**79.61**	79.46
Occupancy Detection	98.66	98.76	98.76	99.15	**99.46**	Failed	99.22	99.27
Phishing	89.84	**91.33**	91.31	90.35	86.4	89.68	90.83	90.64
Rialto	74.69	83.47	81.97	60.78	32.87	Failed	81.42	**86.82**
Room Occupancy	98.31	98.53	98.54	98.9	**99.57**	Failed	99.42	99.52
Spam	88.67	88.72	88.5	DNF	DNF	Failed	94.71	**95.13**

Table 11: **Tail-robust regression MAE with the additional baselines.** All 16 streams; each run's largest 0.1% of absolute errors excluded; means over seeds 1–3, lower is better, best in bold.

Stream	ARF10	ARF100-N	ARF100-U	SRP	SOKNL	ARF-W	ORF
Abalone	1.434	1.403	1.396	1.296	1.754	1.315	**1.285**
Air Quality CO	0.3942	0.3697	0.3825	**0.2535**	0.7901	0.3146	0.304
Appliances Energy	46.95	44.88	53.93	45.83	56.22	33.61	**33.18**
Beijing $PM_{2.5}$	47.64	45.85	47.37	29.9	58.08	27.48	**26.06**
Bike	80.73	76.76	81.92	64.12	87.84	65.56	**61.15**
Demand	22.34	21.77	22.53	21.66	27.9	20.75	**19.5**
Gas Turbine	3.398	3.259	3.323	**2.1**	4.835	2.509	2.412
House8L	17,482	**16,794**	17,738	17,995	19,866	17,564	17,601
Household Power	0.3844	0.3598	0.4097	0.2553	0.3027	0.1592	**0.1484**
Metro	1,573	1,576	1,545	1,730,661	1,595	1,435	**1,281**
NZEnergy	2.148	2.005	2.988	1.844	2.89	1.035	**0.7075**
Online News	2,557	**2,473**	3,221	3,374	3,185	3,210	2,965
Seoul Bike	252.8	244.1	259.3	216.3	327.1	193.9	**192.2**
Steel Energy	3.909	3.411	4.821	4.594	17.17	2.683	**2.506**
Tetouan Power	1,815	1,626	2,091	2,122	4,633	992.8	**991.4**
Trump Approval	0.3174	0.3068	0.3157	**0.1674**	0.4272	0.2867	0.2642

I.4 Ablation details

Table 12 reports the main component comparisons over all 33 datasets. Renewal without challengers removes challengers, tests, and retention checks; no deadline retention permits certification but always renews at the deadline. Uniform vote on ORF trees reuses ORF's physical trees and lifecycle, unlike ARF100-U. Random deadline retention uses ORF's observed rates as retention probabilities (Appendix G), without preserving the realized counts, timing, or identities of retained trees. These comparisons isolate the documented changes within ORF; the comparison of renewal without challengers with ARF-W also changes tree randomness.

Table 12: **Key ablations, pooled over all 33 real-world streams.** Geometric-mean loss ratios (< 1 favors numerator) over the 17 classification (C) and 16 regression (R) streams, reconstructed from rounded summaries.

Comparison	C	R
Renewal without challengers / ARF-W	0.924	0.943
No deadline retention / renewal without challengers (promotion)	0.993	0.995
ORF / no deadline retention (retention checks)	0.988	0.995
ORF / random deadline retention (retention evidence)	0.995	0.996
Uniform vote on ORF trees / ORF	1.532	1.471

Table 13 gives the optimizer comparison and the interaction between weighting and lifecycle. Writing D, W, A, U for ORF, ARF-W, ARF100-U, and uniform aggregation of ORF's trees, the interaction ratio is the geometric mean of $\bar{L}_d^D \bar{L}_d^A / (\bar{L}_d^W \bar{L}_d^U)$. Its value below one indicates that the lifecycle comparison is more favorable under MDW than under uniform aggregation. This does not establish that geometric spacing is optimal or that every replacement improves future loss.

Table 13: **Optimizer comparison and lifecycle–weighter interaction.** Geometric-mean loss ratios over all 17 classification (C) and 16 regression (R) datasets; below one favors the numerator. C uses half-Brier and R uses tail-robust MAE. Except for ORF/ARF-W, computed directly from the primary means, ratios are approximately reconstructed from rounded summaries. The interaction is defined in the text.

Comparison	C	R
ORF (ADER variant) / ORF (MDW)	1.002	1.006
ORF (alternative DR update) / ORF (MDW)	1.116	1.050
ORF (MDW) / ARF-W (MDW)	0.906	0.934
Uniform vote on ORF trees / ARF100-U	1.012	0.963
Lifecycle–weighter interaction	0.895	0.969

The ADER variant and MDW have similar aggregate primary losses. The ADER variant and alternative DR update rows compare the documented implementations, not constants in their theoretical regret bounds. Pooled ablation ratios reconstructed from rounded summaries may differ in their last digit from unrounded run-level aggregates.

I.5 Longer streams and cost

Table 14 checks longer horizons. ORF still trails ARF-W on full Covtype; HTTP's large relative reduction is a small absolute change. NZEnergy's tail-robust MAE ratio moves from 0.684 to 0.736 at 200,000 scored rows. Prefix rankings and gains therefore need not persist at longer horizons. The NZEnergy evaluation does not cover the full source stream.

Table 14: Long-stream sensitivity. Loss and CPU ratios are ORF/ARF-W; losses are half-Brier or tail-robust MAE. Covtype, HTTP, and Demand use full scored streams; NZEnergy stops at 200,000 scored rows. Memory entries are the worst-seed process peaks; host load varied.

Stream	Scored rows	Prefix	Long	CPU	ORF peak	ARF-W peak
Covtype	581,012	1.045	1.018	2.76×	6.7 GB	1.4 GB
HTTP-KDD99	567,498	1.000	0.352	1.72×	0.3 GB	0.3 GB
Demand	149,650	0.940	0.884	1.31×	3.0 GB	0.9 GB
NZEnergy	200,000	0.684	0.736	1.07×	2.3 GB	2.4 GB

Resource comparison under the same load. The resource study reruns the saved configurations of seven methods on all 33 streams, with their catalog horizons and seeds 1–3, in two campaigns of the same protocol: a fixed-seed shuffled queue ran 16 jobs concurrently on the 64-core host, without unrelated jobs, and every rerun reproduced the saved forecasts row for row. The second campaign (25 streams) admitted a job only when the predicted peak memory of the running jobs stayed within 100 GB, so 42 of its stream–method cells include a run with fewer than 12 concurrent neighbours; they are marked in Table 15 and not excluded. A reference ARF-W run repeated seven times across both campaigns kept its CPU time within 2.2%; its elapsed time varied by 16% once, from launcher overhead in the final repetition. Table 2 summarizes task-level ratios; Table 15 gives every stream. Multiclass streams are costly for boosting because SGBT trains one committee per class: on Gas Sensor Drift (six classes) SGBT + correction (5×20) uses 3321 CPU seconds and 22.6 GB, versus 262 seconds and 0.31 GB for ARF-W and 280 seconds and 0.67 GB for ORF. Native boosting was not timed separately, so the cost of SGBT/SGBR + correction includes both learner and head. Long-stream timings used varying host loads. These are implementation-level measurements, including the optimized regression splitter used by ARF-W/ORF, rather than equal-compute comparisons.

Table 15: Resource comparison under the same load on every stream. Ratios of three-seed mean measurements use ARF-W as denominator; memory is peak process memory. The last column names the ratios whose numerator or ARF-W runs include one admitted with fewer than 12 concurrent neighbours under the memory rule of the second campaign (N: ARF100-N, O: ORF, C: SGBT/SGBR + correction).

	ARF100-N / ARF-W		ORF / ARF-W		SGBT/SGBR + correction 5×20 / ARF-W		
Stream	CPU	Memory	CPU	Memory	CPU	Memory	Lighter load
Classification							
Bank Marketing	0.82	0.91	1.39	2.15	0.16	1.59	
Covtype	0.81	0.82	1.42	2.62	3.23	40.27	O, C
EEG Eye State	0.68	1.16	1.91	2.57	0.38	6.73	
Elec2	0.81	1.03	1.37	2.21	0.16	2.56	N, O, C
Gas Sensor Drift	0.94	0.80	1.07	2.13	12.69	72.21	O, C
Hotel Bookings	0.89	0.87	1.83	3.84	0.12	0.65	
HTTP-KDD99	0.59	1.15	1.78	1.00	0.30	4.99	
Image Segments	0.94	1.22	1.08	1.11	1.81	15.50	N, O, C
INSECTS abrupt	0.89	0.86	1.31	2.81	5.01	35.13	O
INSECTS gradual	0.90	0.98	1.31	2.16	4.43	28.17	
INSECTS recurring	0.87	0.79	1.38	3.44	5.39	36.69	N, O, C
NOAA Weather	0.86	0.99	0.94	0.73	0.16	1.55	
Occupancy Detection	0.72	1.17	1.46	1.25	0.27	6.44	
Phishing	0.81	1.20	1.22	0.99	0.60	3.98	O
Rialto	0.89	0.65	1.52	9.13	3.89	33.01	
Room Occupancy	0.71	1.19	1.52	1.15	1.15	9.07	N
Spam	0.95	0.07	1.01	1.09	1.80	6.93	N, O, C
Regression							
Abalone	1.60	1.21	1.42	1.17	0.37	4.55	N, O, C
Air Quality CO	1.46	1.15	1.46	1.35	0.27	4.51	N, O, C
Appliances Energy	1.67	1.06	1.18	1.60	0.19	5.28	N
Beijing PM2.5	1.50	1.01	1.99	3.87	0.16	3.88	C
Bike	1.28	1.11	1.53	1.81	0.17	3.34	
Demand	1.20	1.00	1.16	1.73	0.06	1.17	
Gas Turbine	1.62	0.90	1.81	2.96	0.22	5.60	
House8L	1.18	1.04	1.14	0.76	0.13	1.34	
Household Power	1.35	0.67	2.73	4.95	0.16	2.33	
Metro	1.21	1.04	1.23	1.06	0.13	1.92	N, O, C
NZEnergy	1.26	1.06	1.17	1.12	0.08	1.97	N
Online News	1.29	0.97	0.81	0.56	0.16	1.26	
Seoul Bike	1.38	1.14	1.35	1.49	0.20	4.10	
Steel Energy	1.30	0.93	1.34	1.51	0.18	3.18	
Tetouan Power	1.47	1.03	1.80	2.99	0.18	4.07	N, O, C
Trump Approval	1.26	1.26	1.29	1.00	1.80	4.97	O

I.6 Boosting protocol and per-dataset comparisons

Learner and correction. SGBT/SGBR uses MOA FIMT-DD trees with grace period 50, confidence 0.01, the `-e` option, 75% of features, and OzaBag within each stage. The learning rate is 0.1 for classification and 1 for regression. Binary classification and regression use one committee; multiclass classification uses one per class. Boosting tree counts are per committee.

Algorithms 7 and 8 specify the native anchors, tree features, warm-up/pilot, ADER update, and causal guard. The classification head uses log loss on $[-2, 2]^d$ with pilot G-factor 4. The regression head uses candidate half-squared loss on $[-0.5, 0.5]^d$ with G-factor 1, while evaluation uses the guarded forecast. Warm-up and pilot expose native forecasts and remain in the scores. The guard uses cap multiple 4, moment window 500, scale window 2000, and ridge fraction 0.1. Native boosters and their + correction variants share trained trees; their comparison measures the complete head change, including the regression guard.

Pooled comparisons across shapes. Table 16 pools both 100-tree shapes over all 33 datasets; Table 17 gives every real-world dataset and Table 7 the corresponding classification log losses. At both 100-tree shapes the correction improves native aggregate loss; both forests have lower aggregate classification half-Brier than SGBT + correction, while both SGBR + correction configurations have lower aggregate tail-robust MAE than ARF-W and ORF. Classification rankings also depend on the metric (Table 7). Since the methods also differ in tree implementations, categorical encodings, and numbers of multiclass committees, the loss comparisons do not establish an equal-compute ranking of the model families.

Table 16: **SGBT/SGBR + correction with 100 trees per committee.** Geometric means over 17 classification (half-Brier) and 16 regression (tail-robust MAE) datasets of three-seed mean-loss ratios; below one favors the numerator. C denotes SGBT/SGBR + correction.

Task	Stages × trees	C/native	ARF-W/C	ORF/C
Classification	5×20	0.703	0.966	0.876
Classification	10×10	0.689	0.964	0.874
Regression	5×20	0.889	1.153	1.076
Regression	10×10	0.929	1.158	1.081

Table 17: Boosting with 100 trees per committee on all 33 streams. N: native SGBT/SGBR; C: SGBT/SGBR + correction, including the regression guard. Entries are means over seeds 1–3; lower is better. Regression uses tail-robust MAE in original target units.

Stream	N 5×20	C 5×20	N 10×10	C 10×10	ARF-W	ORF
Classification (half-Brier)						
Bank Marketing	0.09066	0.08705	0.09524	0.09738	0.08139	0.0815
Covtype	0.09146	0.06252	0.09571	0.06692	0.0714	0.07461
EEG Eye State	0.04087	0.00615	0.03845	0.004212	0.007691	0.00609
Elec2	0.1066	0.06187	0.1122	0.05834	0.05504	0.05335
Gas Sensor Drift	0.1104	0.0747	0.1128	0.08009	0.08749	0.05221
HTTP-KDD99	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}	5.0×10^{-6}
Hotel Bookings	0.1203	0.07769	0.1268	0.07117	0.09053	0.06987
INSECTS abrupt	0.212	0.1885	0.2267	0.2111	0.1759	0.1766
INSECTS gradual	0.1893	0.1651	0.2035	0.1836	0.1517	0.1534
INSECTS recurring	0.1887	0.1635	0.2017	0.1825	0.1523	0.1576
Image Segments	0.1008	0.09783	0.1099	0.1086	0.1093	0.08884
NOAA Weather	0.1706	0.1521	0.1852	0.1592	0.1448	0.1452
Occupancy Detection	0.01074	0.005913	0.009547	0.005595	0.006377	0.005877
Phishing	0.09379	0.09254	0.1048	0.1021	0.07077	0.07171
Rialto	0.1194	0.1062	0.1216	0.1142	0.1472	0.1269
Room Occupancy	0.03635	0.02237	0.03208	0.01987	0.004781	0.003843
Spam	0.04927	0.02818	0.04688	0.0241	0.04686	0.04305
Regression (tail-robust MAE)						

Table 17 continued

Stream	N 5×20	C 5×20	N 10×10	C 10×10	ARF-W	ORF
Abalone	1.461	1.44	1.38	1.369	1.315	1.285
Air Quality CO	0.3544	0.3279	0.3396	0.3203	0.3146	0.304
Appliances Energy	38.07	37.16	36.63	35.41	33.61	33.18
Beijing $PM_{2.5}$	27.04	24.06	24.41	22.65	27.48	26.06
Bike	31.25	28.21	31.57	29.89	65.56	61.15
Demand	17.78	17.39	18.09	17.69	20.75	19.5
Gas Turbine	2.337	2.062	2.235	2.089	2.509	2.412
House8L	17,258	17,263	18,383	18,372	17,564	17,601
Household Power	0.2189	0.1945	0.217	0.2052	0.1592	0.1484
Metro	1,379	1,150	1,314	1,114	1,435	1,281
NZEnergy	1.883	1.188	1.389	1.076	1.035	0.7075
Online News	3,038	2,848	3,544	3,097	3,210	2,965
Seoul Bike	160.5	140.4	151.7	139.3	193.9	192.2
Steel Energy	2.064	1.988	2.109	2.093	2.683	2.506
Tetouan Power	896	705.3	854.7	766	992.8	991.4
Trump Approval	0.2862	0.2412	0.2143	0.2036	0.2867	0.2642

I.7 Uncertainty and theoretical scope

Results use three model seeds on the same stream order. Sample SDs describe seed variation, not confidence intervals for paired method differences. The three INSECTS streams share a source and are counted separately, so datasets are not independent replicates.

Source order is preserved, but chronological order is not established for all datasets. Hotel Bookings has unusual ordering, and the source metadata do not establish chronology for NOAA Weather or Rialto. Their results measure prequential prediction in the source order. Table 5 records the sources.

The forest dynamic-regret bound concerns bounded individual-tree losses. Its convexity transfer to prediction loss applies to losses such as half-Brier. Regression feedback is bounded but nonconvex after capping, so the tail-robust MAE comparisons are empirical. For boosting, the convex objectives are classification log loss and unguarded regression half-squared loss. Appendix F accounts for gradient clipping; its candidate-loss bound does not cover the extra regression guard or imply a trimmed-MAE guarantee.

J Conceptual comparison of streaming ensemble components

Table 18 summarizes the modular distinction used in Sections 3–5. It compares mechanisms, not prediction quality or computation. Rows describe the implementations evaluated here (Appendices G, H, and I.3), except AXGB [Montiel et al., 2020], which is included only as a literature comparison. Prediction weights are distinguished from the example weights used during training.

Table 18: **Base learning, prediction weighting, and pool adaptation.** C and R denote classification and regression; online training processes individual examples. The trigger column concerns pool changes, not weight updates.

Method	(0) Base model	(1) Training	(2) Aggregation / weight update	(3) Adaptation trigger	(4) Pool management
Forest methods					
ARF100-N / ARF10 (C/R)	Hoeffding trees	Online bagging; random split features	Accuracy-weighted vote (C); median (R)	ADWIN warnings and drift	Train background trees; replace or reset on drift
ARF-W (C/R)	Hoeffding trees	Same as ARF	MDW tree weights	Same as ARF	Same as ARF
ORF (C/R)	Hoeffding trees	Online bagging; random split features	MDW tree weights; availability mask	Geometric schedule and paired betting tests	Permanent and dynamic slots; challengers; promote, retain, or renew
SRP (C/R)	Hoeffding trees	Online bagging; random feature subspaces	Accuracy-weighted vote (C); uniform mean (R)	ADWIN warnings and drift	Train background trees; replace or reset on drift
SOKNL (R)	FIMT-DD trees; leaf centroids	Online bagging; random feature subspaces	Nearest-leaf mean; self-optimising k	ADWIN; internal tree drift tests	Background-tree replacement; internal subtree adaptation
Boosting methods					
OzaBoost (C)	Hoeffding trees	Online boosting; error-based resampling	Vote coefficients from cumulative weighted errors	No explicit drift trigger	Fixed ensemble
OSB (C)	Hoeffding trees	Online boosting; smooth example weights	Uniform vote (MOA version)	No explicit drift trigger	Fixed ensemble
SGBT/SGBR (C/R)	FIMT-DD trees	Online gradient boosting; bagging within stages	Sum of stage means; no meta-weight update	Internal tree drift tests	Fixed stages; alternate-subtree replacement
SGBT/SGBR + correction (C/R)	Same native trees	Same native training	Native score + box-constrained ADER correction; regression guard	Native tree drift tests	Native lifecycle unchanged
Literature comparison (not evaluated here)					
AXGB (C)	XGBoost trees	Mini-batches; growing, capped windows	Additive boosted scores	Window completion; ADWIN in drift-aware variant	Push or replace trees; drift-aware variant resets window size

Background and challenger trees add physical trees beyond the prediction slots. ARF100-U uses arithmetic means with the same ARF lifecycle; ARF10 uses ten trees. ARF-W and ORF retain slot-weight history across replacements. The boosting correction updates per-tree coefficients in an additive box, whereas MDW assigns nonnegative tree weights. Native ARF regression uses a median, so this comparison does not assume that every method uses the same convex aggregation.

K Adaptation on synthetic regression streams

We examine prequential loss on four synthetic regression streams: stationary, abrupt, gradual, and periodically recurring changes. Each stream has 30,000 observations, 16 features, and a continuous target. Abrupt streams switch among four regimes every 7,500 observations; recurring streams alternate between two regimes every 5,000. Gradual streams continuously interpolate regime parameters, from the first regime to the second over the first half and then to the third over the second half. The methods use the configurations in Appendices I.1 and I.6; boosting has five stages of 20 trees per committee. Figure 2 shows 500-observation trailing mean raw MAE after the 500-observation calibration prefix. Each scenario uses one fixed data stream, with model seeds 1–3 for the three forests and 1–2 for boosting and renewal without challengers. Bands show seed ranges, not confidence intervals or variation across independently generated streams.

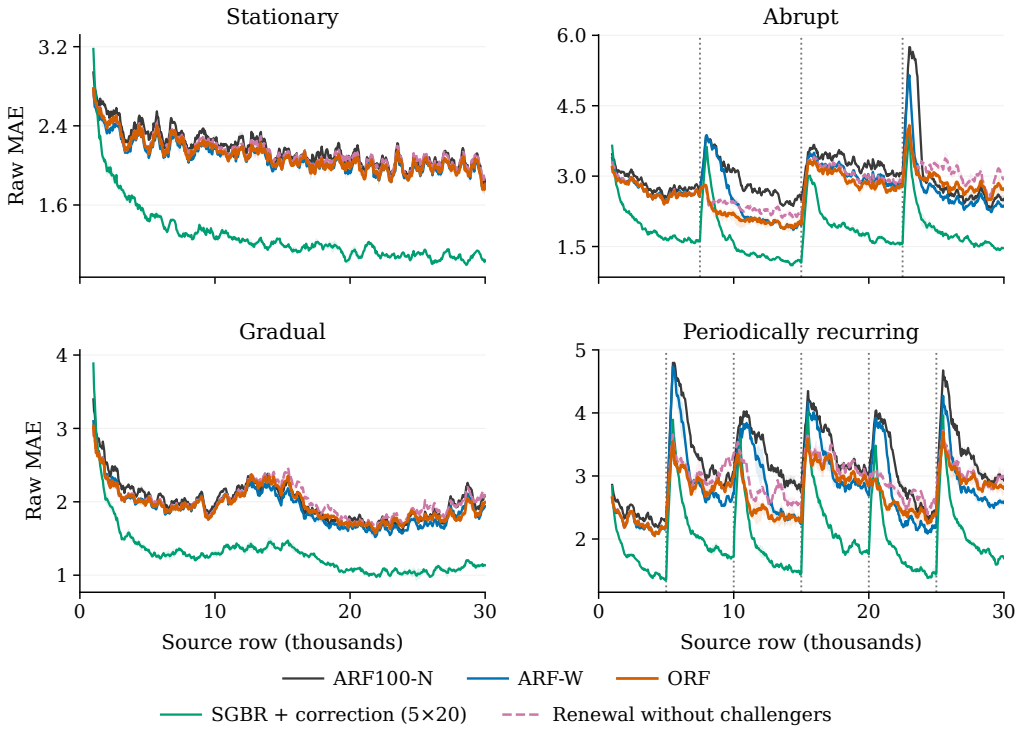


Figure 2: **Regression under controlled changes.** Lines average model seeds; bands show their range. Dotted lines mark abrupt or recurring regime changes; the gradual stream changes continuously. The source observation index includes the unscored calibration prefix.

Post-change loss and whole-stream performance. ORF has smaller immediate loss spikes than ARF100-N and ARF-W after abrupt and recurring changes. Averaged over all change points and three model seeds, its MAE during the first 1,000 observations after each change is 17.5% lower than ARF-W on the abrupt stream and 20.3% lower on the recurring stream. The corresponding whole-stream reductions are 2.7% and 4.2%. Under gradual change, the forest curves remain close and ARF-W has lower whole-stream MAE than ORF.

Renewal and boosting. Renewal without challengers already exhibits much of the reduction in post-change spikes. On matched seeds 1–2, the full ORF lifecycle further reduces whole-stream MAE by 5.2% on the abrupt stream and 4.9% on the recurring stream. SGBR + correction has lower whole-stream MAE than all three forests on all four streams, despite pronounced post-change spikes. These controlled examples illustrate reduced transient loss after abrupt and recurring changes, rather than a uniform advantage across drift types or a guarantee on recovery time.

Data-generating process. Index observations by $t = 0, \ldots, 29{,}999$, and features by $j = 1, \ldots, 16$. Independently over time, draw $z_t \sim \mathcal{N}(0, I_{16})$ and $u_t, \epsilon_t \sim \mathcal{N}(0, 1)$, with independent parameter, feature, and noise generators. Features are

$$x_t = \sqrt{0.7}\, z_t + \sqrt{0.3}\, u_t \mathbf{1}.$$

Each feature has variance one and distinct features have correlation 0.3. Their distribution is stationary; the changes affect the conditional response mean. For regimes $r = 0, \ldots, 3$ and score channels $k = 0, \ldots, 3$, draw $g_{rk} \sim \mathcal{N}(0, I_{16})$ and $b_{rk} \sim \mathcal{N}(0, 0.3^2)$ independently and set $w_{rk} = g_{rk}/\|g_{rk}\|_2$. Regression uses channels zero and one to define

$$f_r(x) = 3.6(w_{r0} - w_{r1})^\top x + 2(b_{r0} - b_{r1}) + \sin(x_1 x_2) - \sin(x_1 x_2 + 1).$$

The response is

$$y_t = \mu_t(x_t) + 0.46(1 + 0.3|x_{t,1}|)\epsilon_t.$$

The stationary, abrupt, and recurring streams use $\mu_t = f_0$, $\mu_t = f_{\lfloor t/7500 \rfloor}$, and $\mu_t = f_{\lfloor t/5000 \rfloor \bmod 2}$, respectively. For gradual change,

$$\mu_t = \begin{cases} (1 - t/15000) f_0 + (t/15000) f_1, & 0 \le t < 15000, \\ (1 - \lambda_t) f_1 + \lambda_t f_2, & 15000 \le t < 30000, \end{cases} \qquad \lambda_t = (t - 15000)/15000.$$

This linearly interpolates the endpoint weights and biases without renormalizing them. The last observed index is $29{,}999$; the hypothetical endpoint at $30{,}000$ is f_2.

Generator seeds and draw order. The data seed is 52917 for every model seed. For each scenario name `stationary`, `abrupt4`, `gradual`, or `recurring`, form a salt by interpreting the first eight hexadecimal digits of SHA-256 of `long_` followed by that name as an integer. NumPy's `SeedSequence([52917, salt]).spawn(3)` initializes three `default_rng` generators, in order for parameters, features, and noise. The parameter generator first draws a standard-normal array of shape $(4, 4, 16)$, normalizes each length-16 vector, and then draws the bias array of shape $(4, 4)$ with standard deviation 0.3. These shapes include the two channels unused by regression. At each observation the feature generator draws the 16 coordinates of z_t and then the scalar u_t; the noise generator draws one ϵ_t. Scenario salts give different parameter, feature, and noise realizations across scenarios, while all methods and model seeds within one scenario share the same stream. The first 500 observations fit the frozen calibration transforms only, leaving 29,500 scored observations.

Stationary examples. We add a low-noise linear stream with 6,000 observations: $x_t \sim \mathcal{N}(0, I_8)$ and $y_t = 4w^\top x_t + 0.12\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, 1)$ is independent and $w = g/\|g\|_2$ for a single draw $g \sim \mathcal{N}(0, I_8)$ fixed throughout the stream. NumPy's `SeedSequence([61907,497600190, 3455138654]).spawn(3)` initializes independent `default_rng` generators for parameters, features, and noise. The first normalized row of a $(2, 8)$ standard-normal draw gives w; the latter generators draw eight feature coordinates and one noise value per observation. The calibration prefix and forest settings are unchanged, with 5,500 scored observations and model seeds 1–3.

Table 19: **Two stationary regression examples.** Whole-stream raw MAE, mean ± sample SD over model seeds 1–3. Each row uses one fixed data stream; lower is better.

Stream	ARF100-N	ARF-W	ORF
Low-noise linear	$\mathbf{0.1183} \pm 0.0009$	0.1652 ± 0.0043	0.1593 ± 0.0011
Nonlinear (Figure 2)	2.1835 ± 0.0100	$\mathbf{2.0879} \pm 0.0197$	2.1056 ± 0.0124

Native ARF100 outperforms both proposed methods on the low-noise linear example, whereas both improve on native ARF100 on the stationary nonlinear stream in Figure 2. Intuitively, under stationarity, ARF and ARF-W may maintain a more stable tree pool because replacement requires a drift alarm, whereas ORF's dynamic slots also undergo scheduled renewal.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims about adaptive weighting and scheduled renewal are supported by the theoretical guarantees in Sections 4–6 and the evaluations in Sections 7–8.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Theoretical assumptions, empirical uncertainty, and computational tradeoffs are discussed in Section 6 and Appendices I.7 and I.5.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Assumptions accompany the theoretical statements; proofs and references to the underlying results are provided in Appendices A–F.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Appendices G, H, and I provide detailed algorithms and experimental protocols.

   Guidelines:

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

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

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

Answer: [No]

Justification: Dataset sources and experimental settings are provided in Table 5 and Appendix I; an openly accessible implementation with reproduction instructions is not provided.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Section 7 and Appendices G–I specify the algorithms, preprocessing, hyperparameters, seeds, and evaluation metrics; Appendix K describes the synthetic protocol.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Table 1 reports means and sample standard deviations over three model seeds on fixed streams; Figure 2 shows seed means and ranges. Their interpretation is described in Appendices I.7 and K.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendices I.1 and I.5 report hardware, CPU time, and memory for the resource benchmark.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research follows the NeurIPS Code of Ethics and evaluates online ensemble methods on existing public benchmarks and synthetic streams.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: The paper focuses on methodological analysis and benchmark evaluation, without a discussion of potential positive and negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained models, scraped datasets, or other assets requiring controlled-release safeguards.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper credits methods and software and identifies dataset sources and software versions (Table 5; Appendix I.1).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new datasets, models, or software assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The study uses existing benchmark data and synthetic streams and does not recruit participants, conduct crowdsourcing, or carry out a new human-subject study.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The study uses existing benchmark data and synthetic streams; no participants are recruited and no participant-facing interventions are conducted.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not components of the core research methods or experimental evaluation.

    Guidelines:

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