International Journal of World Economic Research
ISSN: 3080-9886 (Print), ISSN: 3080-9894 (Online) | Vol. 4, No. 1, 2026

Maturity-Aware Dual-Timescale Replay for Delayed-Feedback Financial Fraud Screening

David Gupta, Ryan White, Kiran Tran *

Department of Computer Science, Florida State University, FL, USA

* **Corresponding author**: (Email: ktran20@fsu.edu)

Abstract: Fraud-screening labels often arrive after a transaction, with fraudulent outcomes confirmed later than legitimate outcomes. This class-dependent observation process distorts conventional replay under concept drift. We instantiate the adaptive recent/global prioritized replay of DriftShield in a one-step cost-sensitive fraud controller, and extend it with maturity-gated access, transaction-time recency and capped inverse confirmation-visibility weighting. A second, arrival-driven scorer supplies a faster adaptation channel; its predictions are combined with the slow replay scorer using a fixed mixture. The system is tested prequentially on an explicitly synthetic 42,000-event stream with two changes in fraud signatures and asymmetric reporting delays. In the three-seed comparison, 42 model/delay runs include three component ablations and two stress-test delays. An additional 12-seed paired evaluation reports F1 0.262 and average precision 0.236 for the dual-timescale controller, versus 0.239 and 0.223 for arrival-time replay. A fast-only scorer with the same threshold controller instead reaches F1 0.265 and average precision 0.242: the slow branch does not improve that stronger comparator in this simulator. Paired intervals and a future-label nonanticipation test delimit the reproducible benefits of maturity-aware replay without implying production deployment gains.

Keywords: Financial fraud detection, delayed feedback, experience replay, concept drift, cost-sensitive policy learning.

1. Introduction

Transaction-level fraud screening poses two distinct challenges: fraud signatures are rare, and they do not stay fixed. Statistical anomaly methods and supervised comparisons have long shown that the information carried by recent transaction behavior can be as important as the model family itself [1, 2, 3]. In an operational stream, however, an immediate alert is rarely followed by an immediate ground-truth label. A cardholder may dispute a payment days later, while other transactions are cleared much sooner. A learner updated only from the labels visible today is therefore trained on a temporally and class-conditionally selected sample, not necessarily on the cohort it has just screened [4].

The distinction becomes critical during abrupt concept drift. The conditional fraud mechanism can change even if the marginal feature distribution appears stable [5]. Approaches that rely on labels becoming available at transaction time silently collapse decision time and supervision time. Delayed outcomes have been analyzed for conversion prediction and online learning [6, 7], and bandit work highlights the role of the outcome-delay distribution [8]. Fraud screening adds asymmetric costs, extreme skew, and finite investigative resources. Adaptive temporal graph neural networks for real-time financial fraud detection provide an informative methodological foundation for measuring how predictive systems respond to evolving fraud patterns [29].

We study the operationally narrow but practically useful case in which every transaction eventually obtains a correct label, but its arrival is delayed in a class-dependent manner. We ask whether a recent/global prioritized replay policy can be made delay-aware without accessing future confirmations. Instead of a large end-to-end actor–critic architecture, we use a one-step reward controller whose policy is a fraud-alert threshold. This reduction isolates the replay and feedback-timing questions while preserving executable action–reward learning.

Our contributions are threefold. First, we define a strictly causal maturity gate for experience replay and an event-time recency window, separating transaction occurrence from label arrival. Second, we propose a bounded inverse-visibility factor and a dual-timescale fast/slow score mixture, with delayed utility controlling the operating threshold. Third, we run multi-seed prequential comparisons and component ablations over known drifting, highly imbalanced synthetic streams, reporting average precision (AP), F1, alarm volume and decision utility rather than a single headline accuracy.

2. Related Work

2.1. Fraud Streams and Supervisory Constraints

Fraud analytics spans outlier detection [9], ensemble classification and hybrid anomaly–supervised strategies [10]. Random forests and gradient boosting have mature implementations [11, 12], while minority-class resampling addresses a different aspect of skew [13]. Their importance as alternatives does not eliminate the challenge of labels arriving after the associated transaction. Dal Pozzolo and colleagues differentiated investigator feedback from delayed labels and designed learning strategies around these separate streams [14]. Streaming active-learning experiments also reveal that selecting promptly reviewed alerts changes the available training evidence [15]. Temporal Transformers combined with conditional tabular GANs for credit-card fraud detection illustrate the value of evaluating sequential structure and minority-class representation together in transaction-risk experiments [30].

2.2. Replay in Non-Stationary Decision Learning

Value-based and policy-based reinforcement learning distinguish policy improvement from the data used to estimate value [16]. Soft Actor-Critic (SAC), for example, is

an off-policy maximum-entropy actor–critic algorithm with continuous actions [17]. Prioritized experience replay samples transitions by a value-residual proxy rather than uniformly [18]. Selective replay can preserve older tasks as a learner encounters new ones [19], while adaptive windows respond to non-stationarity [20]. Our question is more specific: which previously labelled experiences remain admissible and representative when the label process itself is lagged?

2.3. Class-dependent Observability

Conditioning training on observed labels can induce selection bias, a recognized difficulty in learning and evaluation [21]. Importance weighting offers a general correction principle, though classic covariate-shift corrections address a different distributional change [22]. Here the indicator of inclusion is whether a label has arrived by the current decision time. A class-conditional arrival CDF makes this selection mechanism explicit, and the correction is applied only after a transaction's true class has been confirmed. No unreleased class label enters the policy state or model update. Conformal prediction for risk-adjusted financial forecasting highlights the value of reporting uncertainty alongside point predictions, a useful reference for interpreting risk-sensitive decisions under incomplete feedback [31].

3. Problem and Proposed Method

3.1. Delayed One-Step Decision Process

At stream time t, a transaction exposes a feature vector $x_t \in R^d$. The learner emits an estimated risk score p_t and alert $a_t = 1[p_t \geq \tau_t]$. The binary outcome y_t is generated at transaction time but cannot be read by the learner before its confirmation time $m_t = t + D_t$. Denote the visible experience set by $V_t = \{i : m_i \leq t\}$ together with an initially verified historical archive. This gate applies to every update and threshold evaluation. The system chooses actions to maximize observed per-transaction utility, with reward +5 for a detected fraud, −1 for a false alarm, −10 for missed fraud and 0 for a legitimate approval. These numbers are experimental cost units, not estimates of banking losses [23].

$$a_t = 1[p_t \geq \tau_t], \qquad V_t = \{i : i + D_i \leq t\} \tag{1}$$

The simulator is an exogenous one-step environment: transaction outcomes are unaffected by an alert. Accordingly, after y_i becomes visible, the reward of each candidate threshold is identifiable from the recorded score and label. This full-information property permits an interpretable single-step policy improvement rather than an unneeded multi-step credit-assignment assumption. For alert decision a and estimated fraud probability p, the one-step critic is $Q(p,1) = 6p - 1$ and $Q(p,0) = -10p$. The observed temporal-difference residual, with discount factor zero, is $\delta_i = r(a_i, y_i) - Q(p_i, a_i)$.

$$\delta_i = r(a_i, y_i) - Q(p_i, a_i), \qquad \gamma = 0 \tag{2}$$

3.2. Instantiated Two-Tier Replay

Our slow learner explicitly implements the adaptive replay buffer of DriftShield [24], which combines a sliding window of recent experience with prioritized sampling over retained history so that a fraud-screening agent can adapt quickly to drift without forgetting older fraud patterns. Concretely, a fraction β of each minibatch comes from a recent window, the remainder from the full eligible archive, and within each tier transitions receive power-law residual priorities. We retain DriftShield's priority exponent $\omega = 0.6$ and use a recent-tier fraction $\beta = 0.6$; the recent window, a transition count in DriftShield, is redefined below in event time to match the 42,000-event stream. DriftShield's soft actor-critic head, which jointly adapts the alert threshold and feature weights in a continuous action space, is replaced by a transparent one-step value controller over the same threshold decision, while the replay sampling operator remains explicitly instantiated. The slow model is updated by minibatch cross-entropy gradients; the critic residual determines which already confirmed examples enter those updates.

A conventional implementation defines recent experiences by confirmation arrival order. Under delayed positive confirmations that pool may be dominated by legitimate events. We instead define the recent tier by original transaction time: $W_t = \{i \in V_t : 0 \leq t - i \leq 3000\}$. For the baseline replay the recent pool contains the last 1,800 arrivals. Both retain access to the older verified archive, preventing the recent tier from containing only a few observations immediately after a regime switch.

3.3. Feedback-visibility Weighting

Let $F_y(a) = \Pr(D \leq a \mid Y = y)$ denote the confirmation-delay CDF. For a sample i with confirmed y_i and age $A_{it} = t - i$, define $u_{it} = \min\{5, 1/\max[F_{y_i}(A_{it}), 0.05]\}$. The slow learner samples mature example i from a replay tier according to priority $(|\delta_i| + 0.01)^{0.6} u_{it}$. This favors events that would be less likely to have been confirmed by their present age; it never imputes labels for pending events. Selection probabilities are tempered through a bounded inverse-probability factor raised to 0.4 in the minibatch gradient. The numerical caps are deliberate finite-variance controls.

$$P_t(i \mid \text{tier}) \propto (|\delta_i| + 0.01)^{0.6} \cdot u_{it}, \qquad i \in V_t \tag{3}$$

For an event cohort at age a with $F_y(a) > 0$, inverse observability has the familiar identity $E[1\{D \leq a\}/F_y(a) \mid Y = y] = 1$. The actual capped sampler is only an approximation to that ideal correction, and it cannot recover fraud examples at ages when fraud-confirmation probability is zero. The proposed method assumes the class-conditional CDF is known from historical process characteristics; the synthetic experiment uses its configured uniform distributions. Estimating those curves under selective review is a separate deployment problem.

3.4. Fast/slow Integration and Reward Policy

The fast logistic model trains on the 400 most recently arrived streaming labels, using eight gradient steps every 100 transactions. The slow model uses the same update frequency and eight replay steps of batch size 256. Both begin at identical coefficients trained on the first 5,000 confirmed historical records. Predictions are mixed with a fixed 0.6/0.4 fast/slow weight, chosen before examining stream results. The evolving logistic coefficients provide feature sensitivities distinct from the original SAC feature-action weights.

$$p_t = 0.6 \cdot \sigma(w_f^T x_t + b_f) + 0.4 \cdot \sigma(w_s^T x_t + b_s) \tag{4}$$

At each update the alert threshold is re-optimized over a prespecified grid from 0.025 to approximately 0.30, spaced by 0.0125. Candidate action utilities are evaluated only against visible labels, with recency and visibility treatment matching the corresponding method. A smoothed policy

update $\tau \leftarrow 0.85\tau + 0.15\tau^*$ prevents abrupt oscillations. This is a reward-driven contextual control rule; actor updates are carried out by exhaustive one-dimensional policy improvement rather than a stochastic neural policy gradient. The full implementation includes the delayed queue, reward-residual priorities, tier selection, scoring and threshold update. All models share the same simulated transactions and feedback schedule.

3.5. Online Computation and Causal Update Schedule

An update step is deterministic conditional on the seed and its visible experience set. Each new transaction first receives a risk score and a binary decision; its outcome is neither read nor used for the current score. The confirmation queue is then scanned for records whose release time has been reached. Only these records are appended to the eligible archive, along with their transaction indices, realized rewards and residual priorities. After every 100 decisions, the fast and slow parameter vectors are updated from separate views of this same archive. Since the model is refitted after scoring the newest event, a confirmation arriving exactly at time t cannot improve the decision that was already issued at t. This ordering gives a simple audit invariant: every training index at iteration t satisfies $m_i \le t$, except the explicitly verified initial archive.

The recent replay pool is defined by event time rather than by its location in the confirmation queue. We draw 0.6 of the 256 slow-learner examples with replacement from the eligible records no older than 3,000 transaction steps; the remaining 0.4 is drawn from the entire confirmed archive. This mixture is maintained even after a change point when the recent pool contains few fraudulent confirmations. The probability of each eligible record is recomputed from its current residual and its class-conditional visibility. A capped inverse sampling-probability factor enters the gradient to temper the deliberate non-uniform replay. The logistic update uses the gradient of cross-entropy plus a ridge penalty, with batchwise normalization by the total sampling weight, so the effective optimization strength is not proportional to the number of repeated draws of one record.

The fast model and slow model see different evidence distributions but have the same feature representation and loss function. The fast path implicitly tracks release-time recency and makes inexpensive corrections when a batch of new labels arrives. The replay branch continually revisits older, potentially rare outcomes and reduces sensitivity to the latest confirmation batch. Their convex score mixture stays within the unit interval without a second probability calibration model. At each update, the controller tests threshold candidates on labels in the permitted archive and gradually moves its threshold toward the utility-maximizing candidate. This is compatible with a one-step contextual action-value interpretation while avoiding opaque long-horizon returns for an outcome-independent transaction stream.

The implementation uses dense NumPy arrays, and the experimental stream is small enough that the full mature archive can be searched at each replay step. Memory is O(nd) for n events and d features, and the straightforward probability normalization is linear in the eligible archive size. A production implementation would require a bounded ring buffer, sum-tree priority updates or stratified sampling indexes, and stateful recovery of outstanding confirmation messages. These engineering refinements are outside the present empirical question but do not change the causal release rule or the sample-selection equations.

4. Experimental Design

4.1. Data Generation and Timing Integrity

Public fraud datasets do not record when each label was confirmed, so class-dependent confirmation delay cannot be studied on them directly; we therefore generate an explicitly synthetic experiment in which transaction time, ground truth and confirmation time are all known. Each seeded run produces 42,000 transactions with 10 independent standardized Gaussian features. Fraud labels follow a Bernoulli-logistic mechanism with intercept −4.12 and four active coefficients; the dominant three-feature signature rotates from coordinates 0–2 to 3–5 at transaction 15,000, then to 6–8 at 28,000, while a weak ninth-coordinate contribution persists. This induces abrupt changes in P(Y|X) without making the unlabeled marginal feature distribution a giveaway.

Legitimate-label delays are uniform over 25–175 transaction steps; fraud confirmations take 450–1,450 steps. Delay scaling factors 0 and 2 additionally represent immediate and doubled delays. The first 5,000 events form the initially confirmed training archive; only transactions 5,000–41,999 enter prequential evaluation. Future test labels are used exclusively by the evaluator after scoring, never by the learner before their release. Three independent seeds (11, 29, 47) yield evaluated fraud prevalences from 4.72% to 4.99%; this is a designed moderate-imbalance testbed, not a faithful estimate of population fraud prevalence.

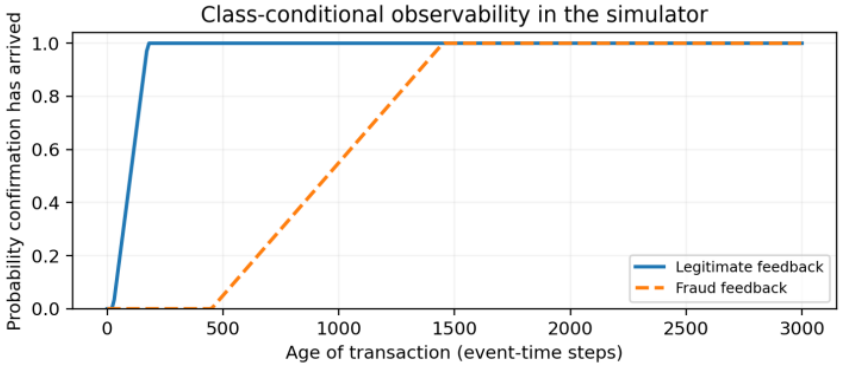


Figure 1. Ground-truth label-arrival CDFs configured in the simulation. Fraud confirmations are systematically slower.

4.2. Comparison Models and Ablations

The comparisons include: (i) a frozen logistic model fitted before live evaluation; (ii) an arrival-only stochastic-gradient learner; (iii) the DriftShield replay baseline [24], i.e., its recent/global prioritized replay with recency defined by label-arrival order; (iv) a single slow delay-aware replay model; and (v) the full fast/slow dual-timescale variant. The ablations remove visibility correction, recent/global mixing, or residual priorities from the single slow replay branch. The replay and online baselines use common learning rate 0.095 and ridge factor 0.0006; initial model fitting and evaluation metrics use scikit-learn [25]. Forest and boosting systems appear in the related-work comparison but not in the executed baseline suite.

Priority refresh uses the discrepancy between realized reward and the one-step critic, and the fast/slow components are not optimized on the held-out future timeline. The global archive contains all previously matured transactions in this 42,000-event experiment; a finite cap beyond 42,000 would leave this experiment unchanged. When an outcome arrives after the final stream event it is recorded as pending, not silently included in earlier updates. No random time-shuffled train/test split is used.

4.3. Metrics and Reporting

Average precision (AP) measures the ranking of rare outcomes under threshold variation and is emphasized for class-imbalanced screening [26, 27]. We also report ROC AUC to describe global ranking [28], precision, recall, F1, alert rate and mean utility per 1,000 decisions. AP and ROC AUC use risk scores generated before any subsequently released feedback; F1 and utility use contemporaneously issued alerts. The original table reports three-seed means, while the accompanying data record contains between-seed standard deviations; neither characterizes production-performance uncertainty. Each ablation and baseline follows the same cost matrix, update frequency and maturation queue.

4.4. Reproducibility Controls and Evaluation Protocol

The generator is defined by an explicit seeded Gaussian random-number stream followed by Bernoulli sampling of logistic outcomes. The same seed selects the features, latent fraud probabilities, realized classes and feedback delays for every method within a delay setting. The two breakpoints alter only the fraud-generation coefficient vector; no learner is given the breakpoint positions as input. Each method is initialized from the same logistic regression estimator trained on the first 5,000 outcomes and the same reward matrix. Methods differ only in their subsequent update stream, replay sampling, propensity adjustment and optional score mixture. This design creates paired comparisons without transferring labels across chronological boundaries.

The threshold grid contains 23 candidates and its smoothing coefficient is fixed to 0.15. The slow replay chooses 256 examples per gradient step, takes eight steps per 100 events and retains an exploratory historical path even when newly released examples dominate recent traffic. The initial three seeds and all numeric settings are saved in the result manifest; nine additional seeds are used for paired robustness testing (Section V-E). Prequential AP is computed from every time-of-decision probability in the 37,000 evaluated transactions per seed; the per-1,000-event curves are secondary diagnostics rather than the source of overall AP. Results from all three seeds are retained, and no run was excluded because a metric was unfavorable.

DriftShield-style replay and the proposed slow replay have comparable parameter counts and update frequencies. The dual-timescale method requires an additional logistic model and corresponding gradient steps, which should be weighed against its F1 gains. Wall-clock comparisons are not reported because execution conditions and multiprocessing contention were not standardized as a systems benchmark. The goal of this experimental design is to identify adaptive accuracy and decision effects without claiming a computational throughput advantage.

5. Results

5.1. Main Comparison

Table I. Three-seed mean prequential performance; between-seed deviations are archived in the result CSV (utility: units per 1,000 events; arrival replay is the DriftShield replay baseline).

Method	AP	F1	Recall	Utility
Frozen LR	0.081	0.129	0.297	-429.6
Arrival SGD	0.246	0.242	0.624	-201.9
Arrival replay	0.226	0.235	0.625	-210.5
Delay-aware replay	0.216	0.249	0.560	-220.5
No vis. correction	0.212	0.256	0.503	-236.8
No two-tier pool	0.132	0.177	0.350	-356.7
No prioritization	0.181	0.228	0.475	-270.1
Dual-timescale (ours)	0.240	0.260	0.563	-209.4

The dual-timescale policy achieves AP 0.240 and F1 0.260, compared with 0.226 and 0.235 for the instantiated arrival-time replay baseline. Its F1 advantage is 0.026 absolute, while the arrival-only SGD baseline obtains the highest AP (0.246) and highest mean utility (-201.9 per 1,000). The result does not support a blanket claim that delayed-aware replay dominates simple online updates; it reveals a tradeoff between stability of action thresholds, alert volume and risk ranking.

The alert rate of the dual controller is 16.1%, versus 21.0% for arrival-time replay and 20.1% for arrival-only SGD. An F1 gain can therefore arise partly from a different alert-operating point rather than exclusively from superior ranking. Reporting AP beside F1 is essential: the delay-aware slow replay alone improves F1 over arrival-time replay (0.249 versus 0.235) while decreasing AP (0.216 versus 0.226).

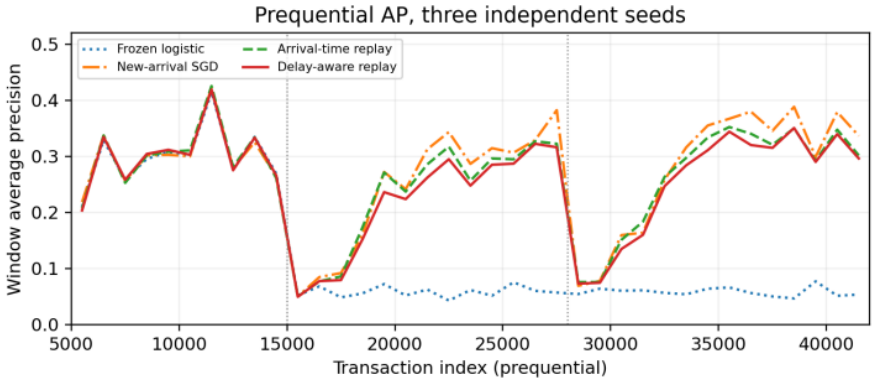


Figure 2. One-thousand-transaction prequential AP windows; dotted vertical markers denote the two true regime boundaries.

5.2. Controlled Component Ablation

The time-aware replay branch has AP 0.216. Removing the inverse-visibility factor changes AP to 0.212, a small descriptive change in all three seeds. Removing the two-tier mixture reduces AP to 0.132; removing residual prioritization reduces it to 0.181. These larger differences show that the dual-pool and residual-priority operations, the two components of DriftShield's adaptive replay [24], contribute materially to the executed replay learner. In F1, however, removing the correction gives 0.256 versus 0.249 for the corrected slow model. Correction favors recall and rankings, not every single-threshold metric.

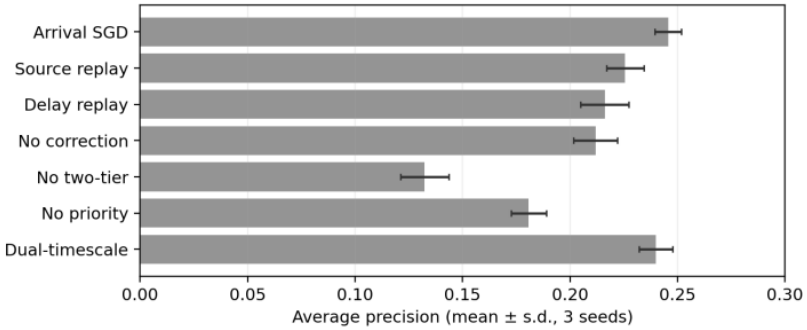


Figure 3. Average precision of the implemented policies and replay ablations; error bars indicate between-seed standard deviations.

5.3. Confirmation-latency Sensitivity

Table II. Feedback-delay stress test; the same drift process is used in each comparison.

Delay	Method	AP	F1
0×	Arrival replay	0.240	0.245
0×	Delay replay	0.232	0.240
0×	Dual timescale	0.258	0.250
1×	Arrival replay	0.226	0.235
1×	Delay replay	0.216	0.249
1×	Dual timescale	0.240	0.260
2×	Arrival replay	0.210	0.226
2×	Delay replay	0.194	0.246
2×	Dual timescale	0.221	0.265

Latency stress tests change only confirmation times, not the generated transactions or their labels for a given seed. AP deteriorates as feedback becomes slower across every tested approach. The dual-timescale variant benefits from maintaining a fast path but remains dependent on newly matured positive examples; it cannot infer an unseen fraud signature before informative confirmations arrive. Under immediate feedback, the visibility correction collapses to unity for all confirmed events. Under doubled delay, the lack of recent positive outcomes amplifies the value of maintaining historical coverage while preserving the limitations of that coverage when signatures rotate.

5.4. Regime-specific Behavior and Threshold Dynamics

The prequential traces in Fig. 2 show a visibly difficult interval after each known switch in the label-generating signature. Unlike a random train–test split, the stream exposes each model to the changing class-conditional boundary in real order. A static fraud signature learned from the initial 5,000 transactions gives poor global ranking after the first switch; the frozen model's overall AP is consequently below that of every updating policy. The online learner benefits as new confirmed outcomes enter the archive, but it cannot observe the first unseen fraud cases before the class-dependent delay expires.

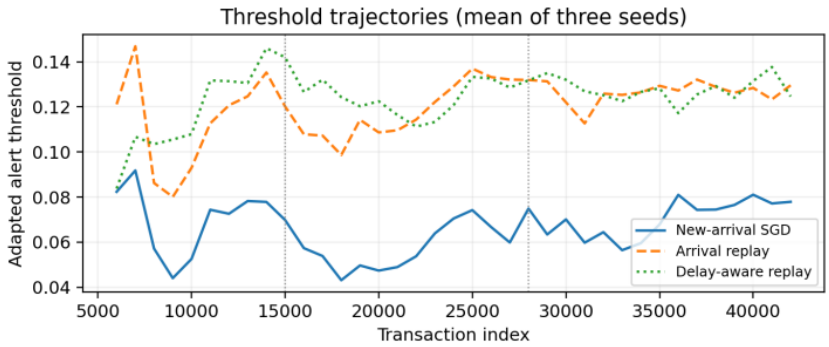


Figure 4. Alert thresholds selected using only feedback available at decision time, averaged over the three seeded runs.

The threshold trajectories in Fig. 4 reveal operating-point adjustments rather than changes in score discrimination. Their movement reflects the cost matrix and the selected replay evidence. A lower threshold may increase recall and alert burden at the same time; a higher threshold may improve precision at the cost of additional missed fraud. This is one explanation for the rankings of methods differing between AP, F1 and utility. The full method obtains a different F1–alert tradeoff under the chosen replay dynamics; another institutional cost ratio could select a different threshold even with unchanged risk scores.

Comparison against the frozen, arrival-driven and replay-only baselines also illustrates why a component improvement should not be confused with global dominance. The full model's average precision gain over the replay-only baseline is evidence that combining fresh and archived evidence is useful under this controlled drift. The fact that the single fast model still has higher AP suggests that replay can also dilute fresh discriminatory evidence when obsolete fraud signatures remain in memory. This tension is particularly sharp when the input marginal stays unchanged and a drift detector based only on unlabeled features has no identifying signal.

5.5. Paired-seed Robustness and Direct Branch Ablation

To test whether the initial three streams were atypical, we reused their records and independently executed nine additional simulator seeds (71, 89, 101, 131, 149, 173, 193, 211, 229) for the arrival-only, DriftShield-style replay, delayed replay and full dual-timescale methods. We additionally executed a fast-only score ablation on all 12 seeds. This ablation sets the mixture coefficient of the slow score to zero while retaining the identical maturity queue, candidate threshold grid and visibility-weighted threshold controller. It tests the value of the slow score, not computational throughput; the audit script retains the otherwise unused slow-path updates.

Table III. Twelve-seed paired differences (dual-timescale minus comparator); t-based 95% intervals describe only the synthetic-generator population.

Comparator	AP difference [95% CI]	F1 difference [95% CI]
Arrival-time replay	+0.0137 [+0.0123, +0.0152]	+0.0229 [+0.0201, +0.0258]
Arrival-only SGD	-0.0059 [-0.0065, -0.0053]	+0.0157 [+0.0133, +0.0180]
Fast-only, matched controller	-0.0059 [-0.0065, -0.0053]	-0.0027 [-0.0046, -0.0008]

On the 12-seed evaluation, arrival-time replay obtains mean AP 0.223 and F1 0.239, while the dual-timescale method reaches 0.236 and 0.262. Their paired gains are consistently positive across all 12 streams. The fast-only scorer and the arrival-only SGD learner produce identical score rankings on every seed, so their AP contrasts in Table III coincide and only threshold-dependent metrics separate them. The direct fast-only comparison changes the conclusion: at the same visibility-weighted threshold policy, removing the slow score increases AP to 0.242 and F1 to 0.265. It also slightly improves reward (-202.3 versus -204.9 per 1,000). Thus the dual controller's F1 gain over the arrival-only SGD baseline is partly attributable to threshold optimization; it is not evidence that slow-score fusion beats a controlled fast-only alternative.

The intervals use the paired standard deviation of per-seed differences with 11 degrees of freedom, without treating 37,000 serially correlated transactions as independent replicates. No hyperparameter search was conducted on the nine added streams. The inference is conditional on this single synthetic generative mechanism and fixed model settings; these intervals do not measure uncertainty about real banking environments. The complete 60-row extended record and 20 paired contrasts are preserved in the reproducibility package.

As an independent leakage check, we executed the replay learner twice on an 8,500-event test stream, changing all unobserved future labels from event 7,100 onward in the second run while leaving earlier inputs fixed. Both pre-intervention 1,000-event windows had identical recorded AP, recall, utility and alert rates. This tests nonanticipation of the implemented replay learner; it is not a complete formal verification of every possible intervention.

6. Discussion and Limitations

The mechanistic comparison helps separate three often-confounded design choices. Replay recency can mean transaction time or confirmation time, which differ substantially when fraudulent outcomes are delayed. Priority by reward residual makes informative but rare fraud outcomes more likely to be revisited. A visibility model corrects for the fact that a just-confirmed fraud represents a lower-probability observation in its event-time cohort. The ablations of Section V-B show that DriftShield's adaptive replay [24] carries most of the slow learner's ranking ability even outside its actor–critic setting, while the smaller change from inverse visibility indicates that the extra modeling assumption introduced here

deserves separate scrutiny.

The dual-timescale combination is deliberately parsimonious. The fast branch tracks the latest confirmed labels and supplies much of the rapid post-drift adaptation; the slower branch broadens historical context. Yet the direct fast-only ablation yields higher AP and slightly higher F1, indicating that broader replay is not uniformly useful for this simulator. This architecture is meaningful even without a multi-step Markov control policy, since risk screening receives exogenous transactions and rewards are attached to individual alert decisions. Reward-sensitive threshold tuning nevertheless requires an explicit valuation of missed fraud and false investigations, which varies substantially across institutions and risk levels.

Several limitations prevent generalization to production fraud detection. The transaction stream is synthetic, has independent standardized feature marginals, and uses abrupt rather than gradual changes in conditional fraud signatures. All eventual labels are assumed correct and all labels eventually arrive; investigator-triggered verification, censoring, action-dependent observations, and reporting errors are absent. The fraud ratio is near 5%, much higher than some public credit-card benchmarks. Delay distributions are given to the correction procedure rather than estimated from a potentially biased operational log. The full ablation/delay matrix spans three independent seeds, while the additional paired robustness comparison spans 12 synthetic streams; neither constitutes independent real-world validation. The grid threshold and mixing ratio were fixed without large-scale hyperparameter search. The one-step policy controller isolates replay decisions from long-horizon SAC credit assignment. These constraints confine the claim to the disclosed simulation and causal timing protocol.

For further validation, a suitable dataset would include at least transaction time, label confirmation time, investigator review actions, and outcomes; otherwise adding artificial delays to a historical fraud table tests a different question. Operational follow-up should estimate F_y jointly with audit policies, evaluate detection at fixed investigative capacity, and compare against fully implemented modern supervised models using time-ordered data. These checks are needed before asserting benefits on financial production streams.

6.1. What the Visibility Correction Identifies

To clarify its statistical role, consider a fixed cohort of transactions of age a. If the label arrival probability conditional on the eventual class equals $F_y(a)>0$, the estimator obtained by weighting each observed outcome by $1/F_y(a)$ has the same expectation as the fully labelled cohort sum, conditional on its generated features and classes. This follows because the label-observation indicator has conditional expectation $F_y(a)$. It is an identification statement about observed cohort totals, not a proof that a sequential non-linear learner converges to a globally optimal fraud decision. Capping the inverse factor at five, using power-law residual priorities, and normalizing each minibatch break exact inverse-probability unbiasedness in exchange for reduced update variance.

The non-positivity boundary is more fundamental. Before the shortest fraud confirmation delay, the probability of observing a positive label in the newly changed cohort is exactly zero in this simulator. No inverse weighting can create information about such latent labels. A method can maintain vigilance via older fraud examples or exploit other unlabeled information, but neither is guaranteed to reveal an entirely new conditional signature. Under doubled delay, this blind interval lengthens and all tested methods lose AP. The reported stress test therefore supports a limited causal conclusion: a feedback-aware replay mechanism can reduce certain observable sampling distortions, but confirmation delay itself remains an information bottleneck.

The present experiments also provide an instructive distinction between label latency and selective supervision. Here every outcome is scheduled to arrive after a known delay law, so the visibility mechanism is independent of the algorithm's alert decision once class is conditioned on. In real case management, analysts may investigate high-scoring alerts first, and negative labels may be inferred through chargeback windows or not appear at all. In that regime, $F_y(a)$ can depend on action, risk score, customer history and the verification policy; an inverse class-only factor is generally insufficient. Evaluation would need recorded logging propensities, audit data or an explicit exploration policy, while ensuring that counterfactual rewards are identifiable for the decisions being compared.

7. Conclusion

We implemented maturity-gated, event-time prioritized replay with bounded inverse label-visibility weighting and examined its use in a dual-timescale fraud controller. Twelve paired synthetic streams confirm better AP and F1 relative to the arrival-time replay implementation, but an equal-threshold fast-only ablation has higher AP and F1. The original three-seed replay ablations support the value of two-tier mixing and residual priority within the replay branch. The chief contribution is an auditable delayed-feedback design and a reproducible analysis of its conditional gains, failure mode, and causal limitations; operational utility remains unestablished.

References

[1] Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235–255. https://doi.org/10.1214/ss/1042727940

[2] Bhattacharyya, S., Jha, S., Tharakunnel, K. K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602–613. https://doi.org/10.1016/j.dss.2010.08.008

[3] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. Data Mining and Knowledge Discovery, 18, 30–55. https://doi.org/10.1007/s10618-008-0116-z

[4] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797. https://doi.org/10.1109/TNNLS.2017.2736643

[5] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 44. https://doi.org/10.1145/2523813

[6] Chapelle, O. (2014). Modeling delayed feedback in display advertising. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1097–1105). https://doi.org/10.1145/2623330.2623634

[7] Joulani, P., György, A., & Szepesvári, C. (2013). Online learning under delayed feedback. In Proceedings of the 30th International Conference on Machine Learning (Vol. PMLR 28, pp. 1453–1461). https://proceedings.mlr.press/v28/joulani13.html

[8] Vernade, C., Cappé, O., & Perchet, V. (2017). Stochastic bandit models for delayed conversions. In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (Vol. PMLR R15, pp. 730–739). https://proceedings.mlr.press/r15/vernade17a.html

[9] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 15. https://doi.org/10.1145/1541880.1541882

[10] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. Information Sciences, 557, 317–331. https://doi.org/10.1016/j.ins.2019.05.042

[11] Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32. https://doi.org/10.1023/A:1010933404324

[12] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). https://doi.org/10.1145/2939672.2939785

[13] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63–72.

[14] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In Proceedings of the International Joint Conference on Neural Networks (IJCNN). https://doi.org/10.1109/IJCNN.2015.7280527

[15] Carcillo, F., et al. (2017). An assessment of streaming active learning strategies for real-life credit card fraud detection. In Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA). https://doi.org/10.1109/DSAA.2017.10

[16] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press. https://mitpress.mit.edu/9780262039246/reinforcement-learning/

[17] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning (Vol. PMLR 80, pp. 1861–1870). https://proceedings.mlr.press/v80/haarnoja18b

[18] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. In Proceedings of the International Conference on Learning Representations (ICLR). https://mlanthology.org/iclr/2016/schaul2016iclr-prioritized/

[19] Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), 3302–3309. https://doi.org/10.1609/aaai.v32i1.11595

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

[21] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In Proceedings of the 21st International Conference on Machine Learning, 114. https://doi.org/10.1145/1015330.1015425

[22] Sugiyama, M., Krauledat, M., & Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. Journal of Machine Learning Research, 8, 985–1005. https://www.jmlr.org/beta/papers/v8/sugiyama07a.html

[23] Elkan, C. (2001). The foundations of cost-sensitive learning. In Proceedings of the 17th International Joint Conference on Artificial Intelligence (pp. 973–978). https://mlanthology.org/ijcai/2001/elkan2001ijcai-foundations/

[24] Cao, J., Zheng, W., Ge, Y., & Wang, J. (2025). DriftShield: Autonomous fraud detection via actor-critic reinforcement learning with dynamic feature reweighting. IEEE Open Journal of the Computer Society, 6, 1166–1177. https://doi.org/10.1109/OJCS.2025.3587001

[25] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html

[26] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

[27] Wang, J., Liu, J., Ge, Y., & Jiang, B. (2023). Risk-aware ensemble learning for credit default prediction under economic uncertainty. World Journal of Economics and Business Research, 1(1), 18–32.

[28] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[29] Yue, X., Yang, J., Zhu, R., & Xu, Q. (2026). Adaptive Temporal Graph Neural Networks for Real-Time Financial Fraud Detection under Concept Drift. Highlights in Business, Economics and Management, 69, 239–250.

[30] Chen, J., et al. (2026). Temporal Transformer with Conditional Tabular GAN for Credit Card Fraud Detection: A Sequential Deep Learning Approach. Mathematics, 14, 1183. https://doi.org/10.3390/math14071183

[31] Chen, Z., Wang, M., Zeng, Z., & Ping, W. (2026). Uncertainty-Aware Financial Forecasting: Leveraging Conformal Prediction for Risk-Adjusted Models. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3702666