Journal of Applied Computational Intelligence

eISSN: 3136-6937

Vol. 4, No. 3, July, 2026

Selective Experience Replay for Workload Drift Adaptive Query Latency Prediction with Long Short-Term Memory

Omar Al-Ghanim[1,*], Tariq Al-Sowaidi[2], Matar Al-Ruwais[2], Hassan Al-Fakhro[1]

[1] *Department of Computer Science, Carnegie Mellon University in Qatar, Doha, PO Box 24866, Qatar*

[2] *Information and Computing Technology Division, College of Science and Engineering, Hamad Bin Khalifa University, Doha, PO Box 5825, Qatar*

Received 28 February 2026; Revised 03 July 2026; Accepted 09 July 2026; Published 14 July 2026

Abstract

Query latency prediction may be inaccurate if there are alterations in the composition of queries, resource contention or other changes to the execution environment. A long short-term memory framework is introduced in this paper to integrate causal sequence encoding and drift-triggered selective experience replay. A bounded memory stores the corresponding observations and preserves the coverage of the previously observed workload regime. The components of replay priority are prediction residual, feature novelty, and retention value; a constrained update controller separates prediction from model adaptation. The evaluation criteria include prediction accuracy, transition response, historical retention, and adaptation cost. Six workload scenarios, six predictors, five memory budgets and five randomization seeds were employed in the reports. The mean absolute error and the 95th percentile of the absolute error for the proposed predictor in the comparison are 8.8 ms and 28 ms, respectively; those for uniform replay are 11.5 ms and 42 ms. The corresponding reductions are 23.5 per cent and 33.3 per cent. After the abrupt transition, recovery delay decreases from 12 to 7 observation windows, with 1,024 memory slots in the proposed configuration. Sensitivity analysis identifies deterioration under excessive replay and limited memory, while recurrence analysis distinguishes adaptation from retention. A framework is provided to test the stability of latency prediction under varying loads. Comparison links memory allocation to the trade-off between rapid adaptation and retention of older workloads in the face of a limited update budget.

Keywords: Selective Experience Replay, Workload Drift, Query Latency Prediction, Long Short-Term Memory, Continual Adaptation

1. Introduction

Machine learning drivers connect predictive models with database steering and optimization tasks [1]. Learned query optimizers rank candidate plans using predicted costs, with estimation errors creating performance-regression risks [2]. Cloud database workloads vary over time in both query composition and load intensity [3]. Concurrent queries interact through shared resources, producing runtime variation across query mixes and system loads [4]. These studies address decisions made before query completion, when execution conditions are partly observable. Historical data capture interactions that are difficult to represent in a fixed-cost expression and can improve prediction accuracy. Changes in operating conditions can affect the relevance of historical observations. A model trained in a low-contention environment may systematically underestimate the latency of the following workload for the same set of submitted queries. The engineering problem therefore has both temporal and structural dimensions. A predictor requires historical context to interpret current requests and a mechanism for adjusting the influence of earlier operating regimes. Continuous

* Corresponding author: omar.ghan@qatar.cmu.edu

http://dx.doi.org/10.68406/jaci.2026.vol4iss3no2:17-32

➢ This is an open access article under the CC BY-NC-SA 4.0 license (https://creativecommons.org/licenses/by/4.0/).

© Authors retain all copyrights.

use of latency estimates links prediction accuracy to subsequent admission and scheduling decisions. Prediction error reflects both representation quality and the relevance of the observations selected for adaptation.

Sequence models represent temporal dependencies in recent service requests and resource measurements. Workload-sensitive learned optimization requires continued adaptation when data distributions change [5]. An LSTM model maintains a compact state representation that captures long-term dependencies in the data. Selection of the historical examples that remain relevant in later updates requires additional memory management. Updating on recent completions can improve adaptation to a new workload while weakening predictions for earlier regimes. Wang et al. [6] examine how continual learning balances the acquisition of new knowledge with the retention of previously learned behavior. Memory capacity, update time, and data retention policies constrain the storage of completed queries. Representative exemplar replay retains selected historical observations to preserve earlier knowledge during incremental updates [7]. Uniform sampling is straightforward, although a bounded sample can underrepresent rare query classes. Sampling solely by prediction error can assign excessive priority to noise and transient outliers. Importance-sampling replay methods use selective buffer construction to retain informative observations and preserve earlier data distributions [8]. The open design question concerns how to balance recency, prediction error, and historical coverage within a fixed replay budget.

This paper addresses query latency prediction when the workload evolves after deployment and completed requests provide delayed supervised feedback. The contribution is an adaptation controller that coordinates causal sequence encoding, drift observation, and bounded replay memory. The recurrent encoder supplies temporal context; the proposed methodological contribution concerns the coordination of update timing, memory admission, and replay frequency. A drift controller determines when an update is justified. A selection mechanism then allocates a fixed replay budget between observations with high current utility and observations retained for coverage. The controller assigns distinct responsibilities to these operations: drift detection initiates adaptation, memory admission preserves useful evidence, and sampling determines training frequency. The intended operating point is a service that can observe submission features, maintain recent resource summaries, and associate a completion with its earlier prediction. Its application to admission or scheduling requires a separate assessment of decision quality under prediction error. The design excludes execution information that becomes available only after the prediction timestamp. It also distinguishes a change in the observable feature distribution from a persistent increase in predictive error. Their joint observation links the timing of adaptation to evidence about both incoming workload composition and the accepted predictor's current reliability. The research question concerns whether this division of responsibilities can improve the balance between adaptation and retention under a fixed replay budget. The evaluation examines this balance across workload transitions while recording prediction error, recovery delay, memory allocation, and adaptation cost under a common sequence of observations.

The remainder of the paper proceeds from limitations in existing models to causal feature construction, controlled adaptation, and evaluation of accuracy, historical retention, and computational cost. Section 2 reviews query performance modeling, sequential adaptation, and replay, identifying the methodological gap addressed by the proposed design. Section 3 introduces the causal prediction task, the sequence representation and the drift observation mechanism. Section 4 specifies memory selection, budget-constrained updates, and operational safeguards. Section 5 presents the evaluation scheme and analysis of prediction error, adaptation behavior, resource requirements and parameter sensitivity. Figures show the aggregated accuracy and scenario-specific trajectories of memory composition under various operating conditions for adaptation decisions. Section 6 reports the conclusions, deficiencies and future work of this study. The replay controller separates the allocation goals of current prediction difficulty and historical representation. The evaluation covers stable operation, abrupt changes, gradual changes, recurrence, burst interference, and mixed transitions. Separate evaluation of these scenarios exposes deterioration on returning workloads and excessive update costs alongside aggregate improvements. This sequence links modeling choices to the information available at prediction time and the criteria used for evaluation.

2. Related Work

2.1. Query Performance Modeling

Query latency models differ in the information available at the time of prediction and the scale of execution description. Learned cost models encode relationships among query plans, data distributions, and runtime behavior [9]. Plan structure, data distribution, and execution conditions describe distinct sources of variation in predicted latency. A plan-oriented predictor can be employed according to the operator structure and estimated cardinalities; a service-oriented predictor also needs to account for queuing and concurrent execution. Comparisons require a common information boundary because models can receive different levels of operator information. The prediction boundary is fixed at submission time, and features use information available at that timestamp. Resource counters are sampled at submission to preserve the causal information boundary. Learned query plan representations encode operator and plan structure to support generalization across related queries [10]. Generalization of related plans is also affected by alterations in the execution environment. A structurally similar query can experience different interference conditions. A fixed-plan identity and altered resource conditions can be used to isolate the cause of degradation. Plan generalization

and adaptation to operating conditions require separate evaluation because structural similarity alone cannot establish robustness to changing contention. A typical submission boundary can also be used to show the effect of different models and adaptation policies under the same information-access conditions.

2.2. Sequential Prediction under Workload Change

Temporal prediction introduces a second dimension of variation. Recent query arrivals, resource occupancy, and completion patterns provide additional contention context beyond the description of an individual request. Recent time-series continual learning methods combine replay with representation alignment to preserve temporal knowledge during adaptation [11]. The sequence encoder preserves the chronological event order. Arrival and completion orders differ; the target latency becomes available for supervised feedback after completion. Resetting hidden states at arbitrary batch boundaries can bias evaluation when test sequences use information unavailable online. Concept drift research distinguishes changes in input distributions from changes in the conditional target relationship [12]. This distinction separates changes in observable workload composition from increased prediction difficulty caused by resource interference. Adaptive-window drift detectors compare statistics across subwindows and resize the retained window after detecting distribution changes [13]. False alarms, detection delay, and prediction accuracy require separate reporting. Event records identify window boundaries and completion times to establish when workload changes become observable. Frequent adaptation can increase scheduling cost and parameter variability. Reporting these effects on a common timeline clarifies how update timing affects model accuracy during stable operation and successive workload transitions. The cited time-series and process-stream studies provide methodological context; their application domains differ from database query latency prediction.

2.3. Replay and Historical Coverage

Experience replay reuses previous observations in subsequent updates. Gu et al. [14] summarize streaming observations into informative replay samples to preserve earlier knowledge under restricted memory capacity. Insertion, eviction and sampling policies determine which operating modes of the predictor will be saved in the replay buffer. Uniform replay gives each stored record the same draw probability. A rare regime therefore contributes few expected draws when it occupies a small fraction of the buffer. Error-based selection gives greater weight to difficult samples, although persistent noise can also receive high priority despite limited learnable structure. Replay quality depends on sample informativeness, representativeness, and diversity across batches [15]. Coverage-based retention preserves representatives of workload regions that occur infrequently in the current stream. The methodological gap is the lack of coordination among these policies with a drift controller and an explicit update budget for query-latency prediction. The value of each design choice depends on its additional storage and training requirements. A controlled comparison requires common observation windows and explicit accounting for memory and scheduled updates. Memory occupancy describes representation within the buffer, while recurrence error measures the predictive effect of that representation. These quantities address separate aspects of retention. The evaluation explicitly specifies recurrence intervals and comparable observation availability across candidate methods. Evidence from general continual learning supports these design choices at the mechanism level; database-specific effectiveness is assessed through the latency comparison.

3. Causal Prediction and Drift Observation

3.1. Prediction Boundary and Feature Construction

The prediction objective is the end-to-end delay of a request from submission to response, including any waiting time in the evaluation stage. Each prediction is based on the sequence of recent arrival events leading up to the current request. Query descriptors are anonymous template identifiers and estimates of plan size and cardinality, operator counts, and a compact encoding of predicates. System descriptors are processor utilization, available memory, storage queue occupancy, and outstanding-request count sampled at submission. Counter availability is set up in the database-specific interface. Missing counters are represented by explicit availability indicators. A transformation fitted during initialization maps the target to a logarithmic scale, and numeric inputs use normalization statistics frozen before evaluation. Unseen template identifiers fall into the category of explicit unknowns and are recorded. The response timestamp marks delivery of the completed result to the submission interface. Queue waiting and execution belong to the same target interval; reporting them separately would require additional instrumentation. Duration units are milliseconds throughout the evaluation.

The recurrent state update and latency readout are coupled in the following equation. The input vector x_t contains the submission features, while h_t and c_t represent the hidden and cell states. The readout parameters w and b transform the hidden representation into a nonnegative latency estimate. A one millisecond offset defines zero duration targets, and bounded readout clipping ensures numerical stability, with clipping counts retained.

$$(h_t, c_t) = \mathrm{LSTM}(x_t, h_{t-1}, c_{t-1}),$$
$$\hat{y}_t = \max\{0, \exp(w^\top h_t + b) - 1\}. \quad (1)$$

Figure 1 describes the causal prediction workflow from query submission through feature assembly, sequence encoding, prediction registration, and delayed completion matching. A separate feedback path carries completed targets into the monitoring and memory interfaces. The workflow explicitly excludes future resource counters from the prediction branch. Sequence context uses arrivals in timestamp order; training records preserve the context available when the prediction was issued. Stable request identifiers order simultaneous arrivals. Restart intervals retain a cold state flag for separate error analysis.

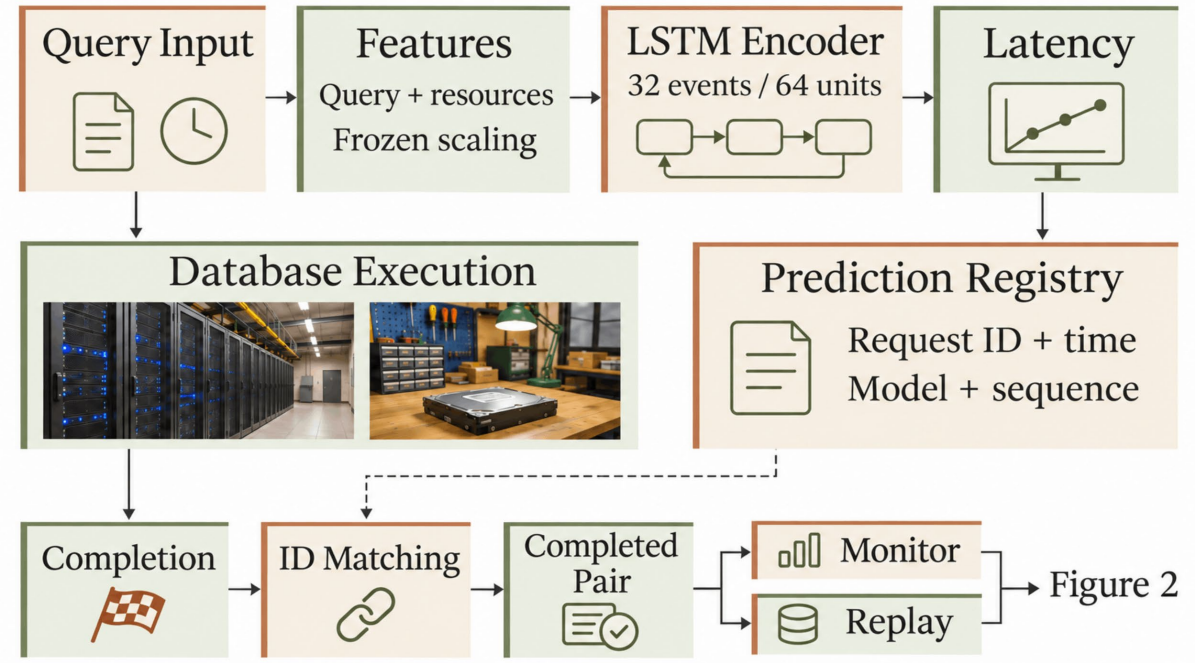


Figure 1. Causal query latency prediction workflow

3.2. Delayed Labels and Error Monitoring

Each outstanding request retains a prediction identifier, submission timestamp, model version, and immutable sequence reference. A completion joins to that record before entering any update batch. The monitor compares each duration with its recorded submission time prediction. Requests canceled before completion are logged separately and excluded from supervised regression unless a censoring model is introduced. Timeout limits are retained as censoring indicators, with completed durations reserved for supervised regression. Completion order may overrepresent short requests within a monitoring window; the monitor therefore reports the number and age of unresolved requests alongside completed errors. A persistent absence of long request labels can delay detection even when the predictor is causal. The reported regression errors consequently characterize requests with observed completion durations. Tail behavior for censored requests requires a separate survival or censoring analysis, and the unresolved-request log establishes the extent of this coverage limitation.

Residuals are calculated in logarithmic space to limit the influence of very long requests. A monitoring window contains a fixed number of matched completions and is compared with a reference distribution established during the latest accepted stable interval. Drift evidence combines residual change with feature distribution change. Feature change is calculated over fixed bins determined during initialization, including an explicit bin for missing values. A large residual statistic with little feature change is retained as a diagnostic because it may indicate an unobserved resource effect. A large feature statistic with stable residuals remains a monitoring event until persistent predictive deterioration provides additional evidence. The drift equation combines these complementary signals through a weighted statistic:

$$d_t = \alpha |\mathrm{med}(R_t) - \mathrm{med}(R_{\mathrm{ref}})| + (1 - \alpha)\mathrm{JS}(P_t, P_{\mathrm{ref}}). \quad (2)$$

Here, R_t and R_{ref} contain absolute logarithmic errors from the current and reference windows. The distributions P_t and P_{ref} describe the corresponding feature frequencies. The residual difference and Jensen-Shannon divergence are scaled using reference dispersion constants established before evaluation. The coefficient α controls their relative influence and lies between zero and one. A positive numerical floor handles zero reference dispersion. The floor and reference distribution remain fixed until the current monitoring decision is complete. Reference replacement occurs only after an accepted stable interval, preserving the distinction between persistent deterioration and stable operation.

Reproducing the numerical drift score also requires the mixing coefficient, dispersion constants, numerical floor, and stable-reference replacement rule. These quantities are configuration inputs; the reported threshold is interpretable only together with their scaling convention.

3.3. Trigger Semantics and Controlled Recovery

A drift decision requires the combined statistic to exceed a fixed threshold for consecutive monitoring windows. The controller uses a threshold of 0.25 and a persistence requirement of two windows. These values specify the controller operating settings. A cooldown interval prevents repeated triggers from creating overlapping update jobs. Cooldown monitoring retains the largest statistic and the full deterioration record. Recovery is defined operationally as the first pair of consecutive windows whose prediction error falls below a prespecified tolerance relative to the new stable regime. The first successful window marks recovery time. This definition distinguishes an isolated favorable batch from sustained recovery, with the subsequent window supplying confirmation. Recovery delay is counted from the transition window to the first successful window of the confirmed pair. Its unit is a completion window, containing 200 matched requests, rather than elapsed wall-clock time.

Threshold selection uses only an initial development stream. The selected threshold remains fixed throughout the chronological evaluation stream. The controller also distinguishes an observed alarm from an accepted update: an alarm can occur while insufficient labels are available or while the update service is busy. Both counts are retained to expose detection activity and operational backpressure. For returning workloads, recovery is measured against the tolerance assigned to that regime during its previous stable appearance, with a documented adjustment if the target distribution itself changes. The monitoring record identifies when adaptation is requested, while the replay mechanism determines which completed observations influence the requested update. Recording these events separately allows recovery delay to be traced to label availability, update scheduling, or candidate acceptance.

4. Selective Replay and Budget Constrained Adaptation

4.1. Memory Admission and Priority

The memory stores completed sequence target pairs together with their submission features, residual at observation, insertion time, and regime assignment. Regime assignment uses coarse clustering of normalized submission features. The configuration uses six such groups and retains an unknown group for unmatched observations. Every admitted example receives a utility score that combines current error, novelty relative to stored examples, and historical retention value. Error clipping limits the influence of extreme durations on priority. Novelty is the distance to the nearest representative under the frozen feature metric. Retention value increases when an example belongs to an underrepresented group or an older regime approaching its minimum coverage quota. The score is evaluated at admission and refreshed lazily when the example is sampled. Novelty therefore describes distance within the available submission-feature representation. A resource change absent from those features can increase residual priority while leaving the novelty score small.

The priority equation assigns separate weights to prediction difficulty, feature novelty, and historical retention:

$$s_i = \beta_e e_i + \beta_n n_i + \beta_r r_i, \beta_e + \beta_n + \beta_r = 1. \tag{3}$$

The components e_i, n_i, and r_i lie in the unit interval after clipping against development stream quantiles. Their default weights are 0.5,0.3, and 0.2, respectively. Development observations determine the candidate configuration. The selection procedure first protects a minimum group quota and then allocates remaining slots by score. If observed groups exceed the available protected capacity, the controller merges the nearest groups and records that operation. A full buffer evicts the lowest scored unprotected example, with insertion order breaking ties. Admission checks quota preservation before evaluating replacement candidates. Consequently, an observation with low current error can remain available because it represents an earlier workload. For the displayed six-group configuration, rounding the 10% reference down gives 102 protected slots per group and 612 slots in total at capacity 1,024. The remaining 412 slots support score-based allocation. Any additional unknown-group reservation must share this fixed capacity and be accounted for explicitly.

Figure 2 specifies the selective replay controller as a feedback workflow connecting monitoring, quota checks, score calculation, candidate selection, bounded updating, and model acceptance. The protected memory and high utility branches expose the respective roles of coverage and sampling. An update job receives a fixed number of completed current examples and replay examples. Both are drawn before optimization begins, and the selected identifiers are logged. Deployment requires a retention policy for sensitive query features. Anonymous descriptors preserve the controller interface while limiting exposure of query details.

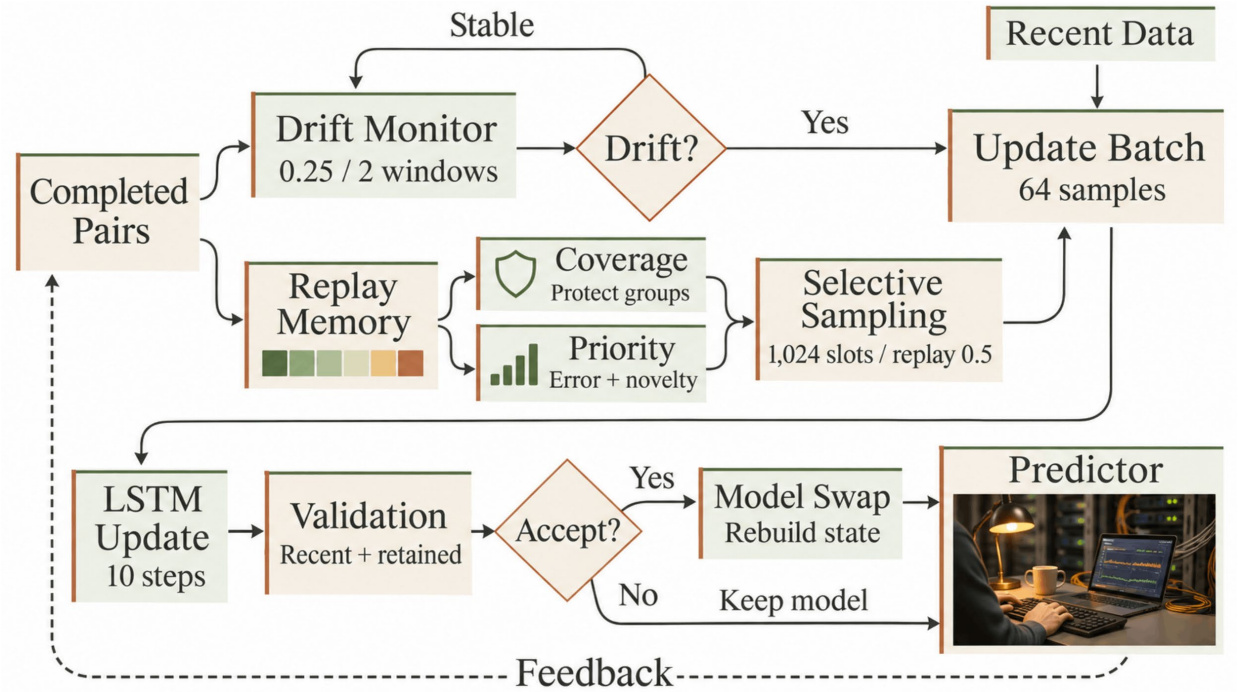


Figure 2. Drift triggered selective replay controller with protected coverage, utility scoring, and bounded model acceptance

4.2. Replay Sampling and Training Objective

Selection into memory and selection from memory have different purposes. Protected quotas determine which historical regions remain available, and replay sampling assigns their influence within each update. A positive floor preserves sampling access for low priority representatives, while a temperature parameter controls concentration toward high scoring records. Replay probabilities follow a tempered sampling equation:

$$p_i = \frac{(s_i + \varepsilon)^{\tau}}{\sum_{j \in \mathcal{M}} (s_j + \varepsilon)^{\tau}}. \tag{4}$$

The set $\mathcal{M}$ contains the available memory records. The default sampling temperature τ is 0.7, and the positive floor ε is 0.01. Sampling with replacement provides directly defined draw probabilities and permits inverse probability weighting. Duplicate draws are recorded, and the memory diagnostic reports the number of distinct examples selected at each concentration level. A variant using sampling without replacement would require the corresponding inclusion probabilities when applying importance correction. Repeated selections count toward the replay batch size, while the distinct-example count describes effective sample diversity. Capped and batch-normalized correction yields a stabilized training objective whose interpretation differs from exact unbiased estimation of a uniform-memory loss.

A current batch and a replay batch contribute to the same logarithmic regression objective. Replay weights are capped and normalized within the sampled batch to limit gradient variance. Capping trades a controlled sampling bias against reduced variance. A parameter proximity penalty discourages large changes from the previously accepted model, while a retention validation check evaluates the effect on protected examples. These requirements enter the training objective through an equation combining current loss, weighted replay loss, and parameter stability:

$$\mathcal{L} = \frac{1}{|\mathcal{C}|} \sum_{i \subset \mathcal{C}} \ell_i + \frac{\lambda}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} v_i \ell_i + \mu \|\theta - \theta_{\text{old}}\|_2^2,$$
$$\ell_i = \text{Huber}\big(\log(1 + y_i) - \log(1 + \hat{y}_i)\big). \tag{5}$$

The current batch $\mathcal{C}$ contains newly completed observations, and the replay batch $\mathcal{B}$ is drawn from memory. The coefficient λ controls replay influence, while μ controls proximity to the previously accepted parameters. Their default values are 0.5 and 0.001. The Huber transition is one logarithmic unit. Inverse probability weights are capped at five before normalization, yielding the batch weights v_i.

An accepted model must reduce validation loss on a chronological holdout of recent completions while remaining within the retention tolerance on protected representatives. Protected validation examples are excluded from gradient updates in that job. The previous model remains available for rollback until the candidate passes the validation gate. Because retained validation observations can become stale, their timestamp range is reported, and their replacement

follows a policy separate from training memory admission. The recent holdout must also remain outside both current and replay gradient batches for that candidate. Its observations become eligible for later training only after their validation role has ended. Reproducing acceptance requires the recent-loss criterion, retention tolerance, holdout size, and replacement schedule.

4.3. Scheduling and Model Acceptance

Prediction continues on the accepted model while an update job trains a candidate copy. The controller limits each job to ten optimizer steps with batch size 64 and one concurrent job. A version swap is atomic, and every outstanding prediction remains associated with its original version. Hidden states require particular care because they were generated under previous parameters. At model replacement, the default implementation reconstructs recent hidden states using the newly accepted recurrent weights. This reconstruction counts toward the adaptation budget. Carrying hidden states across versions offers an approximation whose prediction effect requires separate evaluation, particularly when the accepted update changes the recurrent representation substantially. The optimizer-step limit constrains training work. A wall-clock adaptation allowance additionally covers selection, sequence reconstruction, validation, and version replacement; both accounting boundaries should accompany resource measurements.

Candidate acceptance uses completed holdout observations collected before the job begins. The validation boundary preserves the chronology of candidate construction. An update rejected for poor retention leaves the accepted model unchanged and records both validation losses. Repeated rejection increases underrepresented group diagnostics within the fixed memory and training budgets. Uniform replay and selective replay receive the same update budget throughout the comparison. Recent only adaptation receives the same optimizer steps and current batch size, while frozen prediction retains its initial parameters. Periodic full refresh supplies an operational reference with its historical storage and training effort reported explicitly. The controller records job start time, completion time, validation outcome, and accepted version so that resource consumption can be linked to the predictions served during each transition. The periodic full-refresh reference uses broader historical access and a different training-cost envelope. Its accuracy comparison describes that resource tradeoff; the matched-budget replay comparison concerns uniform, error-prioritized, and selective policies.

5. Experimental Protocol and Numerical Analysis

5.1. Evaluation Design and Prediction Error

The evaluation examines the proposed prediction and adaptation mechanisms under a common set of workload scenarios. It separates prediction accuracy, transition response, historical retention, and update cost so that each operating requirement can be inspected independently. Behavior-model evaluation requires training data that connect database metadata with query-execution telemetry [16]. Accordingly, Table 1 specifies six scenarios: stable operation, abrupt shift, gradual shift, recurring workload, burst interference, and mixed transitions. Each scenario contains 120,000 request positions, allocated chronologically to initialization, development, and evaluation in groups of 24,000, 12,000, and 84,000. Time-series cross-validation evaluates forecasting models through temporally ordered training and validation windows [17]. Five seed labels define the repetition protocol. The comparison includes frozen prediction, recent only adaptation, uniform replay, error prioritized replay, periodic full refresh, and selective replay. All methods share the submission features and completion interface defined earlier. Replay methods additionally share the optimizer step limit and default memory capacity. Continual learning research examines retention of earlier tasks alongside the acquisition of new knowledge [18]. Scenario error, transition recovery, recurrence penalty, memory occupancy, and update cost therefore receive separate treatment. Table 1 also records sequence length, hidden state size, monitoring persistence, replay fraction, and score weights, making the operating point traceable across the comparisons. The initialization prefix ends at request position 24,000, the development prefix at position 36,000, and evaluation starts at position 36,001. A completion becomes eligible for adaptation according to its observation timestamp, preserving the availability boundary when requests cross a partition.

Table 1. Evaluation protocol and model configuration

Item	Specification
Scenarios	Stable, abrupt, gradual, recurring, burst, mixed
Request positions per scenario	120,000; 24,000 initializations; 12,000 developments; 84,000 evaluations
Repeated seeds	Five repetitions; seed estimates and interval ranges
Sequence and network	32 arrival events; 64 hidden units; one recurrent layer
Default memory	1,024 completed sequence target pairs
Monitoring	200 completions per window; threshold 0.25; persistence 2

Item	Specification
Update budget	10 steps; batch 64; replay fraction 0.5
Selection weights	Error 0.5; novelty 0.3; retention 0.2

Figure 3(a) compares scenario mean absolute errors across the six methods. Selective replay gives 6.8, 8.4, 7.9, 8.1, 10.4, and 11.2 ms, whereas uniform replay gives 7.2, 11.1, 10.2, 11.8, 13.9, and 14.8 ms. The absolute difference expanded from 0.4 milliseconds during stable operation to 3.7 milliseconds during recurrence. The larger recurrence contrast is consistent with the intended role of protected coverage. Workload difficulty and update timing also vary across scenarios, so the contrast motivates the component comparison rather than isolating coverage by itself. The burst scenario retains an error of 10.4 ms despite selective replay, indicating a demanding operating condition under the specified workload conditions. Short interference episodes provide relatively little completed supervision before conditions change again. Streaming quantile research treats high-quantile accuracy and bounded storage as joint requirements [19]. Figure 3(b) therefore complements the scenario averages with cumulative error distributions for 600 samples per method. The selective and uniform curves reach their ninety fifth percentiles at 28 and 42 ms, respectively. The 14 ms separation describes the upper error distribution more directly than the 2.7 ms difference between aggregate means. The cumulative curves also expose how much probability mass lies below intermediate tolerances, which matters when prediction error feeds an admission rule. The distribution view complements the scenario averages by locating the error tolerances at which differences between methods are most pronounced.

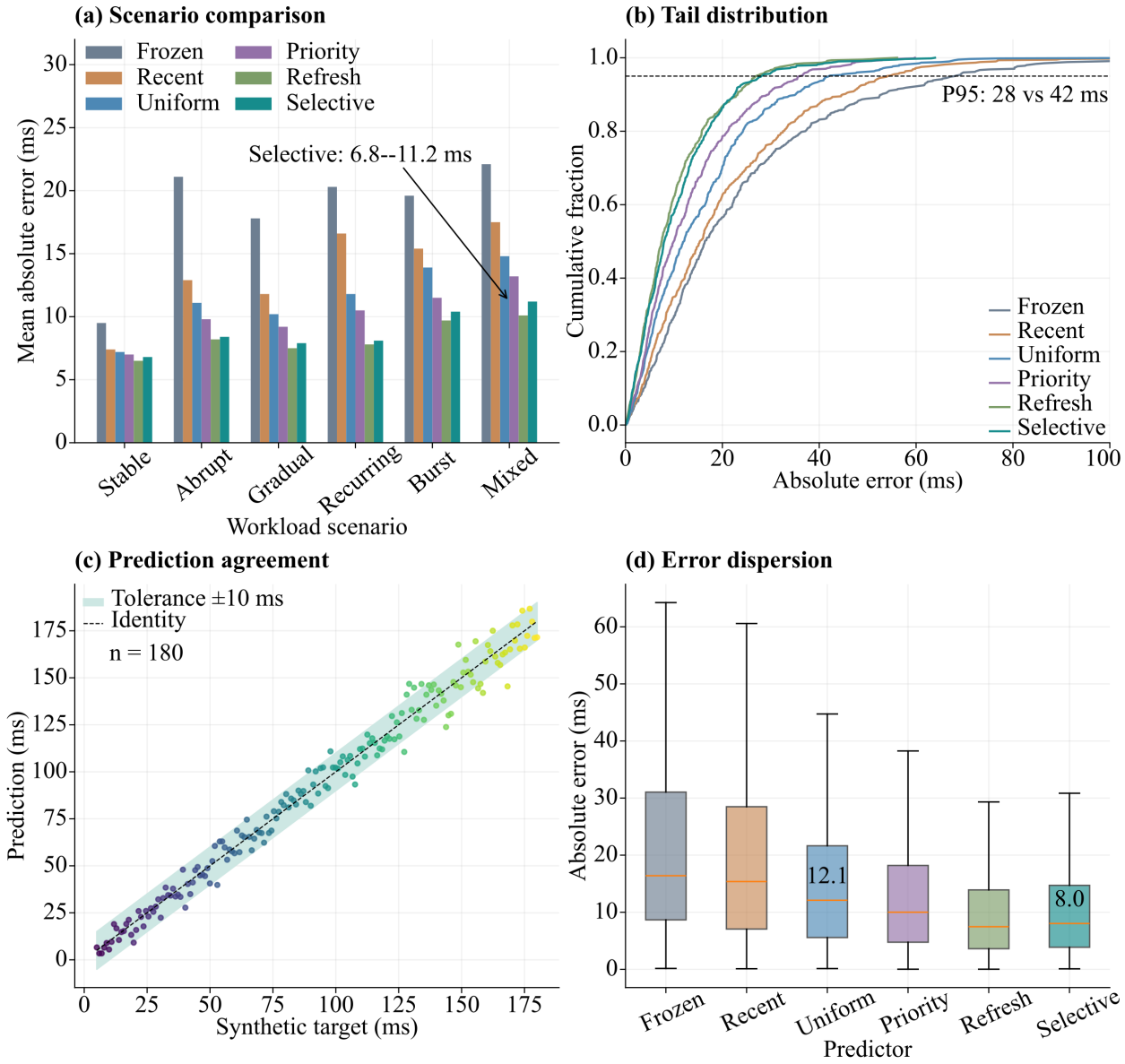


Figure 3. Prediction quality across workload scenarios. (a) Grouped scenario errors. (b) Absolute error cumulative distributions. (c) Prediction agreement with tolerance band. (d) Method error distributions

Figure 3(c) examines prediction agreement for 180 requests spanning target durations from 5 to 180 ms. The identity line marks exact agreement, and the surrounding band represents an absolute tolerance of 10 ms. The residual spread increases with target duration, so long requests occupy a wider vertical range even when their predictions remain close in relative terms. This configuration highlights a practical distinction between absolute tolerance and proportional accuracy: a fixed admission margin becomes more restrictive as conditional variance increases. Comparing the upper and lower sides of the identity line also makes directional error visible. Underestimation can admit more concurrent work than the observed duration would justify, whereas overestimation can reserve capacity unnecessarily. Figure 3(d) presents the associated method distributions as box plots, with selective and uniform median absolute errors of approximately 8.0 and 12.1 ms in the sample distributions. The lower selective median indicates that the error reduction extends into the central error distribution. The box centers, interquartile ranges, and upper whiskers provide complementary information about concentration and occasional large deviations. Squared-error criteria, including RMSE, are used to compare the predictive performance of regression models [20]. For this reason, the scenario average, distribution center, and tail threshold are treated as distinct diagnostics. Their joint interpretation directs attention to the operating tolerance that a downstream scheduler actually applies. Throughout the comparison, P95 denotes the ninety-fifth percentile of absolute prediction error in milliseconds. It describes error magnitude rather than the ninety-fifth percentile of query duration.

Table 2 connects prediction accuracy with adaptation effort. Selective replay has mean absolute error 8.8 ms, root mean square error 14.6 ms, and ninety fifth percentile absolute error 28 ms. Uniform replay has corresponding values of 11.5, 19.2, and 42 ms, giving relative reductions of 23.5%, 24.0%, and 33.3%. The larger tail reduction describes the reported comparison; attribution to individual allocation mechanisms requires the matched component contrasts. Periodic full refresh reaches 8.3 ms mean absolute error at an update cost of 78 ms, while selective replay uses 9.8 ms per job. Recent only adaptation gives 13.6 ms, and frozen prediction gives 18.4 ms with zero online update cost. Confidence intervals for prediction error must account for dependence induced by resampling, since treating fold errors as independent can understate uncertainty [21]. The evaluation protocol therefore specifies retention of paired method predictions for each scenario and seed, together with temporal block identifiers for resampling. Evaluation of learned query optimizers requires explicit training, validation, and test partitions to make comparisons reproducible [22]. Memory capacity, trigger threshold, and replay fraction consequently remain fixed throughout the evaluation suffix. Reconstruction of aggregate entries requires an audit record associating each prediction with its submission time, accepted model version, delayed target, scenario, and seed. For canceled or timed out requests, the record preserves censoring status and observation age so that long running requests remain visible in subsequent tail analysis. The six scenario MAEs have equal weight in the aggregate mean: the listed selective and uniform values average to 8.8 and 11.5 ms. The relative reduction uses the uniform value as denominator. Pooled RMSE and error quantiles require request-level residuals because averaging scenario RMSEs or quantiles changes the statistic.

Table 2. Aggregate prediction accuracy and update job costs

Method	MAE ms	RMSE ms	P95 ms	Update ms
Frozen	18.4	30.1	68	0
Recent only	13.6	23.4	54	6.0
Uniform	11.5	19.2	42	8.1
Error priority	10.2	17.4	36	9.0
Full refresh	8.3	14.0	27	78
Selective	8.8	14.6	28	9.8

5.2. Drift Response and Retention Mechanisms

Figure 4(a) displays the error trajectories around an abrupt transition at observation window 20. Selective replay reaches the 9.5 ms recovery tolerance at window 27, compared with window 32 for uniform replay. The corresponding delays are 7 and 12 windows, giving a reduction of 41.7% under the consecutive window confirmation rule. Both trajectories experience an immediate error increase, followed by different rates of decline. The comparison therefore directs attention to the interval during which the service consumes degraded predictions. Drift-aware neural systems couple change detection with model adaptation to sustain predictive performance on evolving streams [23]. The monitoring record identifies the first observable completion, first alarm, first accepted update, and confirmed recovery as separate events. The frozen trajectory remains above the recovery tolerance throughout the displayed post-transition interval, while recent-only adaptation declines rapidly toward a higher stable error. Therefore, the first reduction in error may be achieved at the expense of a worse stable-state operating point. A shaded transition interval and tolerance line are used to show the two behaviors at different times. Recovery labels identify the first successful window, with the

following window supplying confirmation. Long-running requests stretch a given number of completion windows over a longer elapsed interval, making completion ages necessary for interpreting recovery delay.

Figure 4(b) isolates the relationship between recovery and historical retention. Selective replay combines a 7-window recovery delay with a recurrence penalty of 1.4 ms. Uniform replay occupies 12 windows and 3.8 ms, while recent only adaptation occupies 6 windows and 7.2 ms. Recent only adaptation thus has the shortest delay among these three configurations, but the largest deterioration when a previous regime returns. The selective policy allocates part of the update to earlier observations, linking its longer initial adjustment to a smaller recurrence penalty in this comparison. This allocation exposes the tradeoff between rapid recovery and retention of earlier workload behavior. Periodic full refresh has a recovery delay of 5 observation windows and a recurrence penalty of 1.1 ms; its larger memory marker represents broader historical access. The position needs to be considered along with the update cost in Table 2. The frozen predictor is plotted at the 30-window censoring boundary with a recurrence penalty of 10 ms. The horizontal coordinate represents the observation limit for unresolved recovery; its actual recovery delay exceeds that limit. Marker areas represent logical memory allocations, while the vertical coordinate measures prediction deterioration. Separate encodings allow memory allocation and retention performance to be assessed independently. Recurrence comparisons repeat the earlier query and arrival trace to preserve the target conditions needed to attribute the penalty to predictor changes.

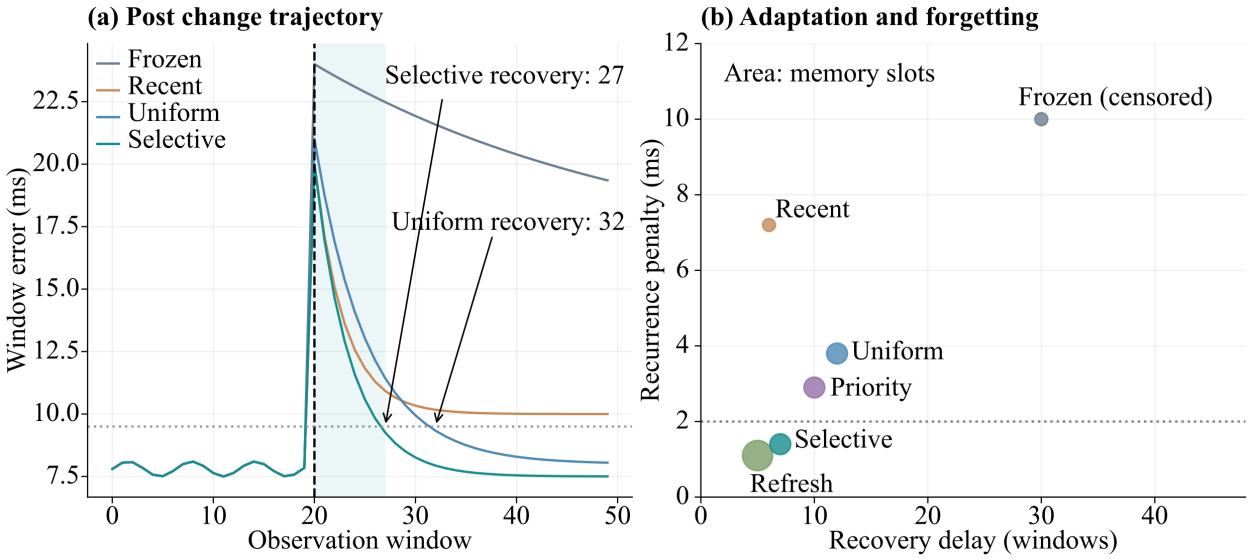


Figure 4. Drift response and retention tradeoff. (a) Window error after an abrupt change. (b) Recovery delay versus recurrence penalty with memory scaled markers

Component analysis in continual learning can distinguish mechanisms that preserve previously learned representations under replay imbalance [24]. Figure 5(a) specifies six such comparisons. The complete selective design has mean absolute error 8.8 ms; removing coverage increases it to 10.3 ms, removing error priority to 10.0 ms, removing novelty to 9.7 ms, and disabling the drift trigger to 10.8 ms. Recent only adaptation reaches 13.6 ms. The 1.5 ms coverage contrast targets returning workload representation, while the 1.2 ms error priority contrast targets allocation toward currently difficult observations. Their different roles explain why a single high error sampling rule may leave part of the retention problem unresolved. The 0.9 ms novelty contrast represents the value assigned to feature regions that are poorly represented in memory. The trigger contrast is larger, at 2.0 ms, because the timing of an update determines which completions are available and which accepted model serves the transition. For component isolation, the candidate acceptance gate, optimizer steps, feature normalization, and chronological observations remain matched across ablations. Logging accepted and rejected candidates identifies whether error differences arise from memory contents, sampling frequency, or validation decisions. Keeping these records also allows an expensive but repeatedly rejected update policy to be identified from its operational history. These contrasts evaluate component removal within the stated controller. Interaction among coverage, sampling, and the acceptance gate limits interpretation of each difference as an independent additive contribution.

Figure 5(b) follows the memory composition of six groups over eight checkpoints at the default capacity of 1,024 slots. Group 1 decreases from 25% to 11%, while Group 5 increases from 12% to 20%. The remaining groups absorb the changing allocation, with protected occupancy maintained at or above the 10% reference throughout the checkpoint sequence. At this capacity, a 10% share corresponds to approximately 102 representative records, giving the visual reference a concrete storage interpretation. The gradual redistribution reflects a policy that preserves older regions while admitting newer observations into the remaining capacity. Occupancy and replay influence remain separate records because a stored example can receive infrequent sampling. Delayed-label regression reuses historical labeled

observations while incorporating recent inputs whose targets remain unavailable [25]. Therefore, a temporary increase in the replay share of a protected group may be due to delayed current completion or intentional retention. The occupancy log associates each checkpoint with admission counts, eviction counts, group age, and replay draw frequency. The log also records outstanding long requests because delayed completions affect which groups appear difficult at update time. A stress case alternates a high-latency template between repeatable contention and irreducible noise, then examines score clipping, group occupancy, and selection frequency under the same quota policy.

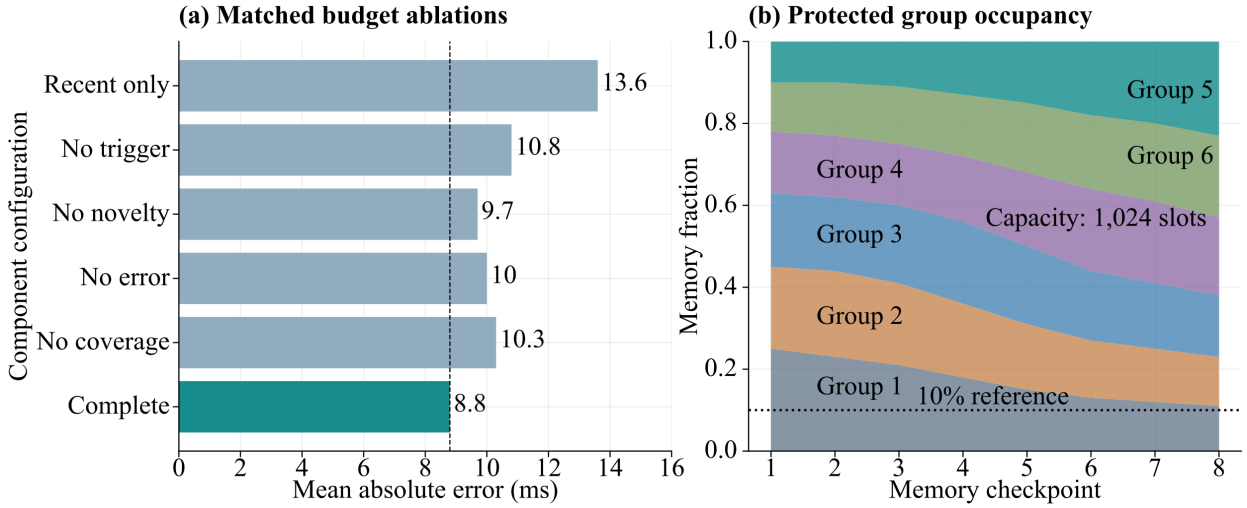


Figure 5. Mechanism diagnostics. (a) Component ablation errors. (b) Protected memory group occupancy across checkpoints

5.3. Resource Sensitivity and Operating Boundaries

Elastic database resource management must account for shared-resource contention and workload-dependent capacity requirements [26]. Figure 6(a) presents update job distributions around reference costs of 6.0 ms for recent only adaptation, 8.1 ms for uniform replay, 9.0 ms for error priority, 78 ms for full refresh, and 9.8 ms for selective replay. The 15 ms boundary distinguishes configurations whose reference job costs fit the declared adaptation allowance. Selective prediction also has an inference reference of 0.34 ms per request. The distribution widths represent variation around each positive job cost, directing attention to occasional expensive scoring and validation operations. Frozen prediction has zero update cost and is identified separately at the display floor of the logarithmic axis.

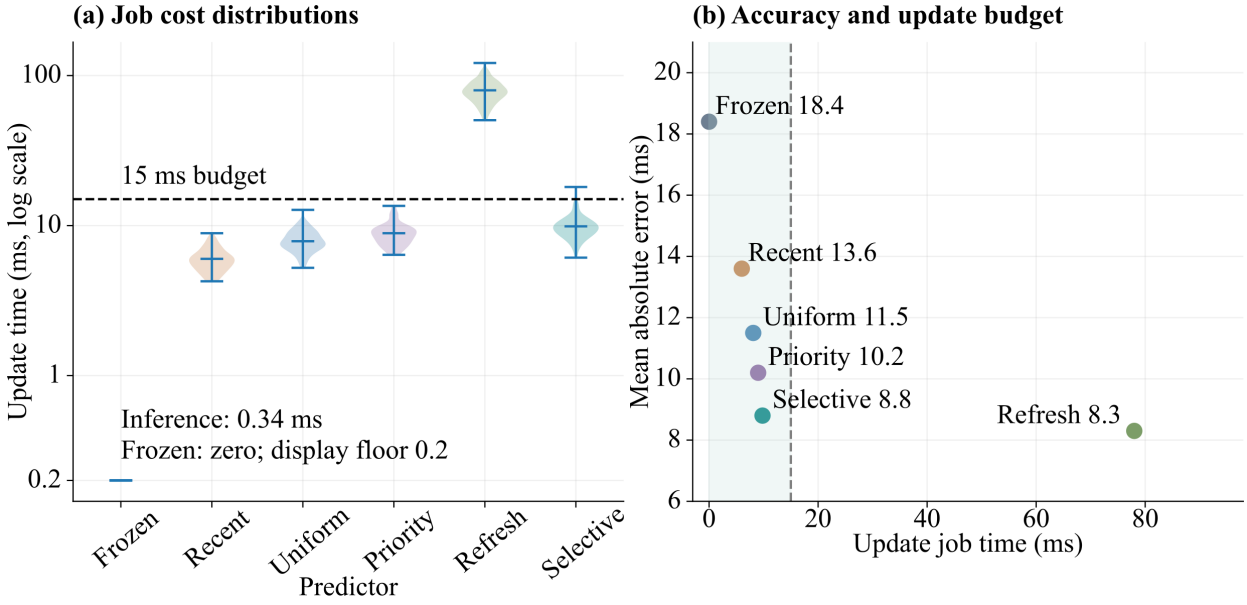


Figure 6. Resource requirements. (a) Update time distributions and inference reference. (b) Error versus update time under a declared job budget

Figure 6(b) pairs the same update costs with aggregate errors. Relative to uniform replay, selective replay spends an additional 1.7 ms per update job for a 2.7 ms reduction in mean absolute error. Full refresh gains a further 0.5 ms in accuracy while increasing the job cost by 68.2 ms. This comparison locates the reported accuracy and job-cost tradeoff

relative to the declared allowance. Deployment transfer additionally requires the processor and accelerator models, concurrency, software stack, warm-up policy, and timing boundaries. Resource accounting should include memory scoring, sequence reconstruction, candidate validation, and model replacement in the update path. These operations can contend with prediction even when optimization executes asynchronously, especially when arrival intensity rises during the same transition that initiates adaptation. The 0.34 ms reference is per prediction, whereas 9.8 ms is per selective update job; combining them into a service overhead requires the number of predictions and update jobs over a common elapsed interval.

Figure 7(a) examines five memory capacities: 256, 512, 1,024, 2,048, and 4,096 slots. Selective replay has corresponding errors of 12.4, 10.5, 8.8, 8.4, and 8.2 ms. Increasing capacity from 256 to 1,024 slots reduces the error by 3.6 ms, whereas the next fourfold increase reduces it by 0.6 ms. The selected operating point therefore lies near the change in marginal benefit, where additional historical coverage yields progressively smaller improvements. Uniform and error prioritized replay remain above the selective curve across the five budgets, preserving a matched capacity comparison. Hyperparameter optimization selects model configurations using a defined evaluation procedure and explicit computational objectives [27]. Figure 7(b) extends that selection problem to five replay fractions and five drift thresholds. The displayed grid minimum of 8.8 ms occurs at replay fraction 0.50 and threshold 0.25. At the same threshold, fractions 0.10 and 0.90 give 11.0 and 11.0 ms, respectively. The symmetric penalty at the extremes represents two allocation pressures: limited historical influence at low replay and reduced attention to incoming observations at high replay. At fraction 0.50, thresholds 0.10 and 0.50 give 10.2 and 10.8 ms. An overly sensitive controller expends updates on short fluctuations, while a less sensitive controller waits longer for accumulated evidence. The surface specifies the joint contrasts that a development stream should evaluate under a fixed optimizer budget. This finite-grid minimum characterizes the inspected operating points. Configuration selection uses the development prefix, and the evaluation suffix serves only to report the selected configuration and prespecified sensitivity contrasts.

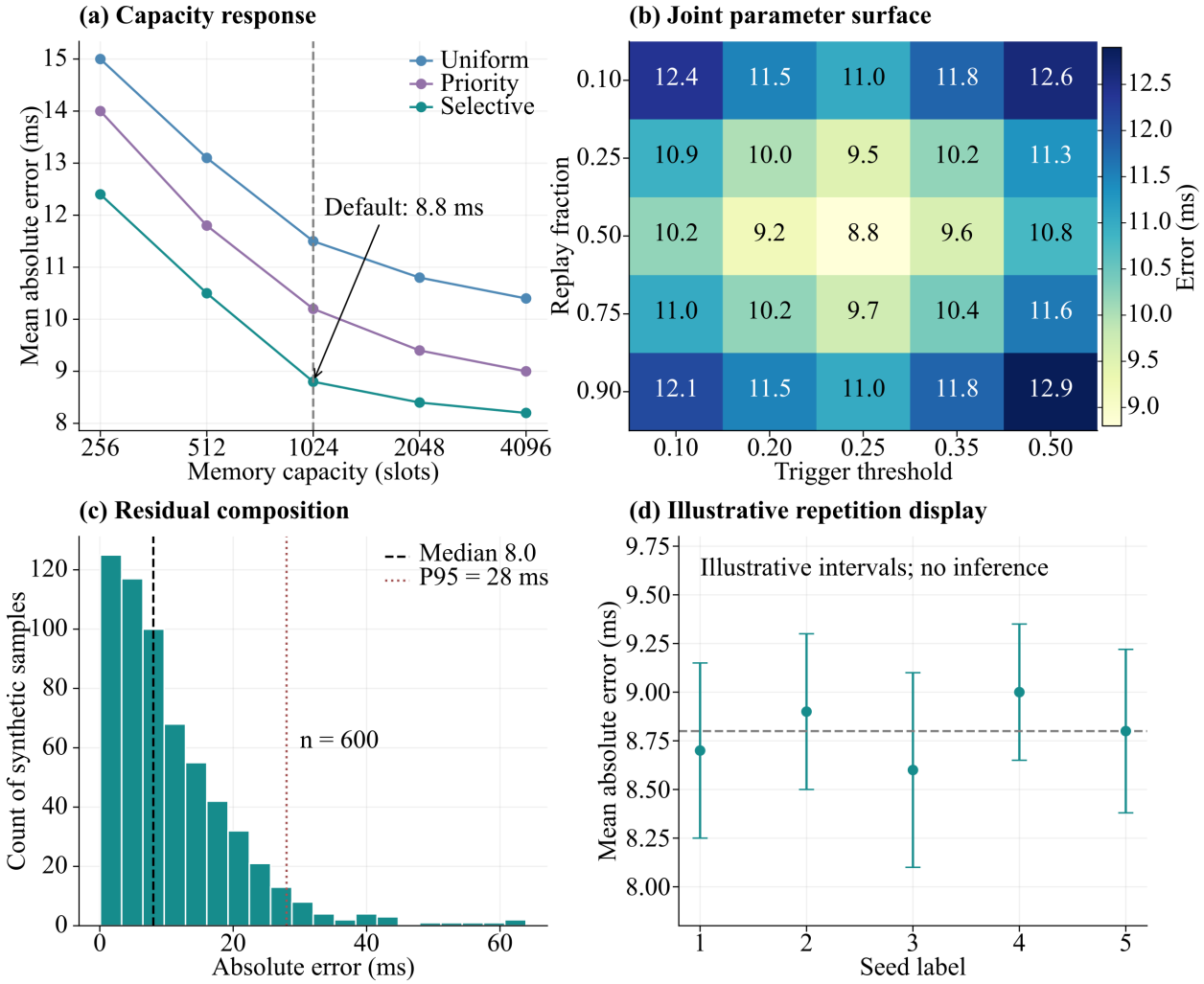


Figure 7. Sensitivity and uncertainty diagnostics. (a) Memory capacity response. (b) Replay fraction and trigger threshold heatmap. (c) Absolute error histogram. (d) Seed estimates and interval ranges

Figure 7(c) shows the error composition of 600 selective replay residuals. Their median is approximately 8.0 ms and their ninety fifth percentile is 28 ms. The dense lower bins represent the majority of requests, while the thinner upper

bins preserve large deviations that influence tail oriented operating tolerances. Both reference markers retain their positions relative to the histogram bins, making the distinction between central concentration and tail extent explicit. Bin counts also reveal how few observations support the outer range. This feature becomes consequential when overlapping request sequences coincide with correlated error bursts. Figure 7(d) presents five seed estimates, centered at 8.7, 8.9, 8.6, 9.0, and 8.8 ms. Their displayed interval half widths range from 0.35 to 0.50 ms. These ranges provide descriptive context for the seed centers. Inferential coverage requires a specified confidence level, block-selection procedure, and paired run records; visual overlap alone establishes neither equivalence nor a significance decision. Interval estimation requires paired run records and temporal blocks that preserve the dependence among adjacent observations during resampling. Seed variation is also examined separately for recurrence and burst interference, where the order of delayed completions can alter both the memory admissions and the first accepted update.

Feature-drift analysis identifies distribution changes across correlated input features [28]. Figure 8 examines the relationship between observable shift severity and recovery through 12 transitions whose normalized shift magnitudes span 0.10 to 0.90 and whose recovery delays span 3 to 16 windows. Transition T5 has magnitude 0.40 and delay 7 windows, while T6 has a greater magnitude of 0.48 but a shorter delay of 6 windows. Their ordering indicates that feature-divergence magnitude alone is insufficient to rank the displayed recovery delays. Label composition and completion timing are candidate explanations that require event-level checks. The largest transition, T12, combines magnitude 0.90, delay 16 windows, and a recurrence penalty of 3.4 ms. Bubble areas show a protected replay share rising from 20% to 50% across the cases. Resource-constrained continual learning evaluates replay quality together with storage requirements and computational overhead [29]. A larger protected share can preserve historical influence while reducing the fraction of updates devoted to a genuinely novel regime. That pressure becomes stronger when long current requests remain unresolved and the available completion pool is dominated by earlier workload conditions. The seven-window reference separates moderate transitions from cases requiring extended adjustment. For transitions above that line, the event log should retain outstanding request ages, novel template frequency, score refresh age, and validation rejections. These measurements distinguish insufficient current supervision from selection policies that continue to allocate too much influence to historical examples during repeated rapid changes.

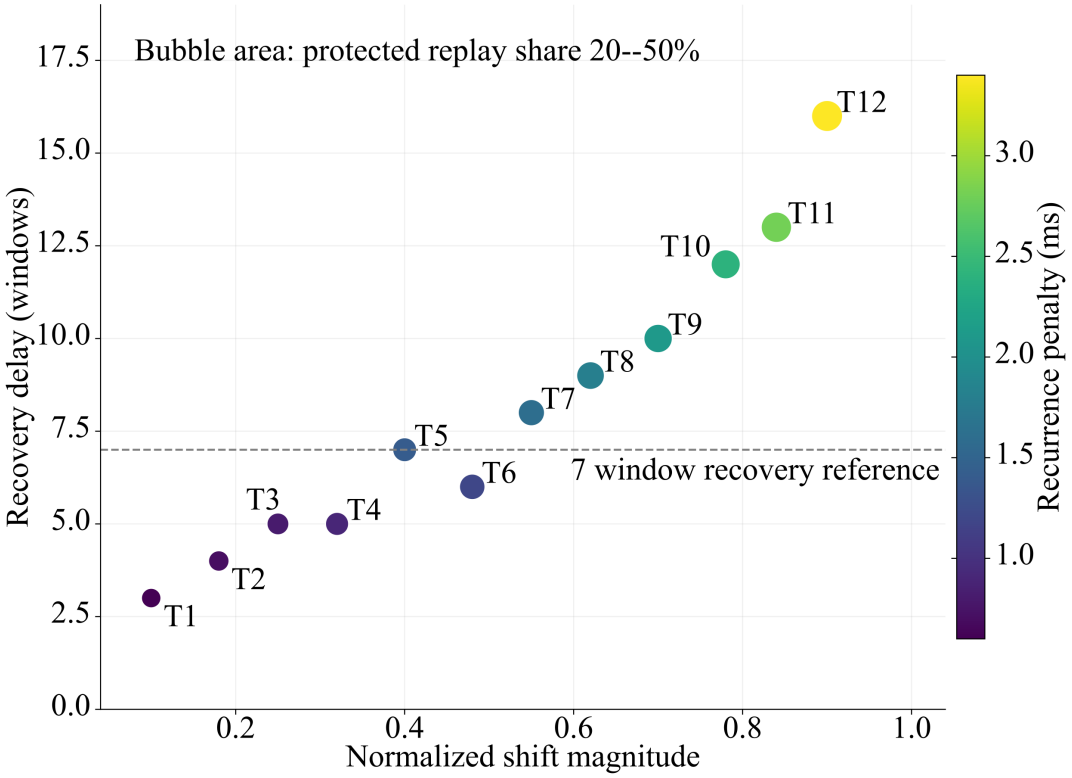


Figure 8. Transition severity and recovery delay with replay allocation and recurrence penalty encodings

6. Conclusion

This paper developed a workload-adaptive query latency prediction framework based on long short-term memory and selective experience replay. The design separates causal prediction, drift observation, memory coverage, and bounded model updating. Replay priority combines prediction difficulty, feature novelty, and historical retention. Protected admission preserves workload coverage, and the validation gate assesses candidate models before acceptance. The evaluation examines prediction accuracy, recovery, retention, and computational cost. These mechanisms operate within a specified adaptation budget.

The present study has several limitations. The operating scope depends on causal submission features, reliable completion matching, and sufficient update capacity during workload transitions. The model also assumes that the submission features and delayed-completion labels can be combined reliably. Hidden resource effects, censored requests, evolving query representations, and stale validation examples can reduce the controller's efficiency. The quality of the grouping procedure for protected memory coverage will determine how many examples are still predictive of the target. The effect of parameter proximity and validation gates depends on how well the retained observations represent returning workloads. Reproducibility also depends on reporting the complete drift scaling, holdout configuration, timing environment, and uncertainty procedure. These details determine how precisely the numerical comparison can be independently reconstructed.

Future research should extend the controller across database environments with different query execution models, resource counters, and completion delay characteristics. Particular attention should be given to delayed supervision, rapid successive transitions, and the cost of updating while prediction remains active. Further research can be conducted on adaptive group formation, uncertainty-aware admission, and retention policies for sensitive query descriptors. A reproducible implementation should retain timestamps, model versions, and replay histories to trace performance degradation to observation delays, memory selection, or candidate acceptance. These studies will identify the applications of selective replay and determine when simpler adaptation policies are more suitable.

7. Declarations

7.1 Author Contributions

Omar Al-Ghanim contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision, project administration, and funding acquisition. Tariq Al-Sowaidi contributes to validation, analysis, investigation, data collection, draft preparation. Matar Al-Ruwais contributes to conceptualization, validation, analysis. Hassan Al-Fakhro contributes to investigation, data collection. All authors have read and agreed with the manuscript before its submission and publication.

7.2 Funding and Acknowledgments

This research received no specific financial support from any funding agency.

7.3 Institutional Review Board Statement

Not applicable.

7.4 Informed Consent Statement

Not applicable.

7.5 Date Availability

Not applicable.

7.6 Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7.7 Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors utilized Quillbot for linguistic polishing, logical organization and draft revision. The generative AI tool was not involved in research design, data collection, data analysis, or the equation of core conclusions. All outputs generated by AI were reviewed, revised and verified manually by the authors. The authors take full responsibility for all content of this manuscript.

8. References

[1] Zhu, R., Weng, L., Wei, W., Wu, D., Peng, J., Wang, Y., Ding, B., Lian, D., Zheng, B., & Zhou, J. (2024). PilotScope: Steering databases with machine learning drivers. Proceedings of the VLDB Endowment, 17(5), 980–993.https://doi.org/10.14778/3641204.3641209

[2] Weng, L., Zhu, R., Wu, D., Ding, B., Zheng, B., & Zhou, J. (2024). Eraser: Eliminating performance regression on learned query optimizer. Proceedings of the VLDB Endowment, 17(5), 926–938. https://doi.org/10.14778/3641204.3641205

[3] van Renen, A., Horn, D., Pfeil, P., Vaidya, K., Dong, W., Narayanaswamy, M., Liu, Z., Saxena, G., Kipf, A., & Kraska, T. (2024).

Why TPC is not enough: An analysis of the Amazon Redshift fleet. Proceedings of the VLDB Endowment, 17(11), 3694–3706. https://doi.org/10.14778/3681954.3682031

[4] Wu, Z., Markakis, M., Liu, C., Chen, P. B., Narayanaswamy, B., Kraska, T., & Madden, S. (2025). Improving DBMS scheduling decisions with accurate performance prediction on concurrent queries. Proceedings of the VLDB Endowment, 18(11), 4185–4198. https://doi.org/10.14778/3749646.3749686

[5] Ramadan, M., Mokhtar, H. M. O., Sobh, I., & El-Kilany, A. (2025). Towards online training for RL-based query optimizer. International Journal of Data Science and Analytics, 20(3), 2711–2720. https://doi.org/10.1007/s41060-024-00628-4

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

[7] Zhang, X., Chen, Y., Ma, C., Fang, Y., & King, I. (2024). Influential exemplar replay for incremental learning in recommender systems. Proceedings of the AAAI Conference on Artificial Intelligence, 38(8), 9368–9376. https://doi.org/10.1609/aaai.v38i8.28790

[8] Li, J., Azizov, D., Li, Y., & Liang, S. (2024). Contrastive continual learning with importance sampling and prototype-instance relation distillation. Proceedings of the AAAI Conference on Artificial Intelligence, 38(12), 13554–13562. https://doi.org/10.1609/aaai.v38i12.29259

[9] Heinrich, R., Li, X., Luthra, M., & Kaoudi, Z. (2025). Learned cost models for query optimization: From batch to streaming systems. Proceedings of the VLDB Endowment, 18(12), 5482–5487. https://doi.org/10.14778/3750601.3750699

[10] Lyu, C., Lachaud, G., Lozano, G., & Diao, Y. (2025). Graph transformers for query plan representation: Potentials and challenges. Proceedings of the VLDB Endowment, 18(13), 5716–5730. https://doi.org/10.14778/3773731.3773745

[11] Yin, T., Wang, J., Wang, C., Wang, H., Cai, J., Liu, M., Ma, Y., Gao, K., Song, Y., & Shen, W. (2026). R-Tuning: Wavelet-decomposed replay and semantic alignment for continual adaptation of pretrained time-series models. Proceedings of the AAAI Conference on Artificial Intelligence, 40(33), 27870–27878. https://doi.org/10.1609/aaai.v40i33.40010

[12] Lukats, D., Zielinski, O., Hahn, A., & Stahl, F. (2025). A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams. International Journal of Data Science and Analytics, 19(1), 1–31. https://doi.org/10.1007/s41060-024-00620-y

[13] Pasquadibisceglie, V. (2026). Handling concept drifts with traditional process discovery algorithms. Journal of Intelligent Information Systems, 64(1), 179–213. https://doi.org/10.1007/s10844-025-00955-3

[14] Gu, J., Wang, K., Jiang, W., & You, Y. (2024). Summarizing stream data for memory-constrained online continual learning. Proceedings of the AAAI Conference on Artificial Intelligence, 38(11), 12217–12225. https://doi.org/10.1609/aaai.v38i11.29111

[15] Krutsylo, A. (2024). The inter-batch diversity of samples in experience replay for continual learning. Proceedings of the AAAI Conference on Artificial Intelligence, 38(21), 23395–23396. https://doi.org/10.1609/aaai.v38i21.30398

[16] Lim, W. S., Ma, L., Zhang, W., Butrovich, M., Arch, S., & Pavlo, A. (2024). Hit the gym: Accelerating query execution to efficiently bootstrap behavior models for self-driving database management systems. Proceedings of the VLDB Endowment, 17(11), 3680–3693. https://doi.org/10.14778/3681954.3682030

[17] Lai, M., Wulff, S. S., Cao, Y., Robinson, T. J., & Rajapaksha, R. (2025). Temporal cross-validation in forecasting: A case study of COVID-19 incidence using wastewater data. Quality and Reliability Engineering International, 41(2), 672–688. https://doi.org/10.1002/qre.3686

[18] Li, S., Su, T., Zhang, X.-Y., & Wang, Z. (2025). Continual learning with knowledge distillation: A survey. IEEE Transactions on Neural Networks and Learning Systems, 36(6), 9798–9818. https://doi.org/10.1109/tnnls.2024.3476068

[19] Fan, Z., Cai, Y., Dong, S., Yin, Q., Bai, T., Xue, H., Chen, P., Wu, Y., Yang, T., & Cui, B. (2025). Per-flow quantile estimation using M4 framework. IEEE Transactions on Knowledge and Data Engineering, 37(8), 4758–4775. https://doi.org/10.1109/tkde.2025.3573812

[20] Emami, S., & Martínez-Muñoz, G. (2024). Deep learning for multi-output regression using gradient boosting. IEEE Access, 12, 17760–17772. https://doi.org/10.1109/access.2024.3359115

[21] Bates, S., Hastie, T., & Tibshirani, R. (2024). Cross-validation: What does it estimate and how well does it do it? Journal of the American Statistical Association, 119(546), 1434–1445. https://doi.org/10.1080/01621459.2023.2197686

[22] Lehmann, C., Sulimov, P., & Stockinger, K. (2024). Is your learned query optimizer behaving as you expect? A machine learning perspective. Proceedings of the VLDB Endowment, 17(7), 1565–1577. https://doi.org/10.14778/3654621.3654625

[23] Chambers, L., Gaber, M.-M., & Ghomeshi, H. (2025). DeepStreamEnsemble: Streaming adaptation to concept drift in deep neural networks. International Journal of Machine Learning and Cybernetics, 16(5–6), 3955–3976.

https://doi.org/10.1007/s13042-024-02492-x

[24] Xu, S., Meng, G., Nie, X., Ni, B., Fan, B., & Xiang, S. (2024). Defying imbalanced forgetting in class incremental learning. Proceedings of the AAAI Conference on Artificial Intelligence, 38(14), 16211–16219. https://doi.org/10.1609/aaai.v38i14.29555

[25] Chen, L., Wu, G., Dong, H., & Hao, K. (2026). A novel adaptive soft sensing framework for label delay in industrial data streams. ISA Transactions, 174, 164–174. https://doi.org/10.1016/j.isatra.2025.09.007

[26] Barnhart, B., Brooker, M., Chinenkov, D., Hooper, T., Im, J., Jha, P. C., Kraska, T., Kurakula, A., Kuznetsov, A., McAlister, G., Muthukrishnan, A., Narayanan, A., Terry, D., Urgaonkar, B., & Yan, J. (2024). Resource management in Aurora Serverless. Proceedings of the VLDB Endowment, 17(12), 4038–4050. https://doi.org/10.14778/3685800.3685825

[27] Tan, J. M., Liao, H., Liu, W., Fan, C., Huang, J., Liu, Z., & Yan, J. (2024). Hyperparameter optimization: Classics, acceleration, online, multi-objective, and tools. Mathematical Biosciences and Engineering, 21(6), 6289–6335. https://doi.org/10.3934/mbe.2024275

[28] Szűcs, G., & Németh, M. (2025). Domino drift effect approach for probability estimation of feature drift in high-dimensional data. Knowledge and Information Systems, 67(5), 4597–4621. https://doi.org/10.1007/s10115-025-02362-0

[29] Kim, J., & Kim, S. (2025). Salient frequency-aware exemplar compression for resource-constrained online continual learning. Proceedings of the AAAI Conference on Artificial Intelligence, 39(17), 17895–17903. https://doi.org/10.1609/aaai.v39i17.33968