Volatility-Driven Decay: Adaptive Memory Retention for RAG Systems Under Unknown Drift

Abe Diaz
Independent Researcher
`https://github.com/abe238/volatility-driven-decay`

February 2026

Abstract

Retrieval-Augmented Generation (RAG) systems face a silent reliability crisis: when knowledge drifts—APIs deprecated, guidelines revised, frameworks migrated—retrieval continues serving outdated answers with full confidence. We find that naive retrieval serves stale content 67% of the time across 120 real-world technical facts. Existing solutions assume known drift patterns: recency-biased retrieval suits constant change, time-weighted decay handles gradual transitions, and sliding windows work for bounded contexts. None adapts when the pattern is unknown—the typical production condition.

We introduce Volatility-Driven Decay (VDD), where the memory decay rate $\lambda(t)$ dynamically adjusts based on detected semantic drift via sigmoid activation on embedding centroid distance. Structurally inspired by control-theoretic adaptive filtering and cognitive models of memory consolidation, VDD continuously modulates its forgetting rate between a conservative baseline and an aggressive ceiling. Validated on 120 facts across React, Python, and Node.js documentation ($n = 30$, bootstrap CIs, Benjamini–Hochberg corrected, Cohen's d), VDD achieves 87.5% error reduction versus static baselines, ranks #2 among 13 methods across four drift patterns, and is never the worst performer in any tested scenario.

An auto-calibration method (75th percentile of burn-in volatilities) eliminates manual V_0 tuning, improving over hand-tuned defaults by 22.7% on average.

We are equally explicit about limitations: recency outperforms VDD in permanent high-drift ($d = -2.90$), time-weighted retrieval dominates gradual transitions (0.871 vs 0.584), and adaptive baselines (Holt–Winters, EMA-lambda) beat VDD in 3/4 synthetic scenarios. On StreamingQA (36K temporal QA pairs), all decay methods including VDD underperform no-decay ($d = -7.68$), confirming that VDD is counterproductive on knowledge-accumulation tasks—it is specifically designed for knowledge-replacement scenarios. On FreshQA (452 temporal questions), all decay methods achieve 100% current-answer retrieval rate versus no-decay's 70–80%, though keyword-overlap scoring fails to capture this advantage. Rankings transfer between hash-based and real neural embeddings (4/5 scenario agreement, $r = 0.935$ effect size correlation) and across embedding models (nomic-embed-text vs mxbai-embed-large, Spearman $\rho = 0.978$), though effect sizes attenuate with real embeddings (mean ratio 0.70x), indicating controlled-condition results are directionally valid but inflated. VDD's value is precisely for production systems where drift patterns cannot be predicted in advance. This positioning complements—rather than competes with—modern RAG architectures (Self-RAG, CRAG, Adaptive-RAG) that improve *what* to retrieve, while VDD addresses *when to forget*. We release the complete benchmark, dataset, and evaluation framework.

Keywords: RAG, memory decay, concept drift, adaptive retrieval, knowledge staleness, LLM memory

1 Introduction

1.1 The Knowledge Staleness Problem

Retrieval-Augmented Generation has become the dominant architecture for grounding large language models in external knowledge [30, 15]. Yet RAG systems harbor a failure mode the community has largely ignored: **knowledge staleness**. When the world changes—React deprecates lifecycle methods, Python removes a module, a medical guideline is revised—retrieval systems continue surfacing outdated answers. The user receives a confident, well-formatted, *wrong* answer.

This is not hypothetical. In our evaluation of 120 real-world facts across three documentation ecosystems, naive retrieval (no decay) serves stale content **66.7% of the time**. Even commonly recommended approaches fail: sliding window retrieval catastrophically collapses to random-chance accuracy (0.333) on certain domain structures.

1.2 The Static Decay Dilemma

The "Forgetting is a Feature" hypothesis [50] established that biologically inspired decay improves retrieval precision. However, existing implementations rely on **static decay**—a fixed half-life applied uniformly regardless of environmental conditions:

- **Conservative decay** ($\lambda \sim 0.1$) preserves valid knowledge but accumulates stale facts (IAE 148, nearly 5x worse than adaptive approaches).
- **Aggressive decay** ($\lambda \geq 0.5$) stays current but catastrophically forgets when knowledge reverts (IAE 52.1 vs VDD's 35.4 in reversion scenarios).
- **No fixed rate handles both.** A coding assistant may face gradual API deprecation for months, then a sudden breaking change—within the same deployment.

1.3 The Modern RAG Landscape

Recent architectures have advanced retrieval quality along multiple axes. Self-RAG [3] introduces reflection tokens for retrieval-time quality control. CRAG [52] adds corrective retrieval to handle low-quality results. Adaptive-RAG [24] routes queries by complexity. FLARE [25] triggers retrieval actively during generation. RAPTOR [46] builds hierarchical document summaries for multi-level retrieval.

These architectures improve *what* and *how* to retrieve. None addresses *when to forget*—the temporal validity of stored knowledge. VDD operates at the storage layer, determining how aggressively to discount old memories based on environmental change. This is orthogonal and complementary: a Self-RAG system with VDD-managed memory would both select high-quality passages *and* automatically deprioritize stale ones.

1.4 Our Proposal

We propose that the optimal decay rate is not a constant but a function of **environmental volatility**. When the semantic landscape is stable, decay should be slow. When embeddings shift—signaling knowledge change—decay should accelerate. We ground this in control-theoretic adaptive filtering, drawing structural inspiration from Recursive Least Squares (RLS) where a forgetting factor dynamically weights past observations, and in cognitive models of hippocampal memory consolidation where environmental novelty accelerates forgetting of outdated associations [51, 11].

1.5 Contributions

1. A **drift detection method** using embedding centroid distance achieving 100% recall with 0.12ms latency.
2. **Proof of adaptive behavior**: VDD's λ is genuinely bimodal (Hartigan's Dip $p < 0.001$, Ashman's $D > 2$), not noisy static decay.
3. **Multi-domain validation** on 120 facts across React, Python, and Node.js with real embeddings (nomic-embed-text).
4. **Never-worst robustness**: VDD ranks #2 among 13 methods across all four tested drift patterns.
5. **Activation function ablation** showing all tested activations converge at recommended $k \geq 5$ ($d \to 0$), justifying sigmoid selection on interpretability grounds.
6. **Hash-to-real embedding transfer validation** demonstrating ranking preservation across embedding types (4/5 scenarios, $r = 0.935$).
7. **Rigorous multiple comparison correction**: 85/88 significant results survive Benjamini–Hochberg FDR correction with Type M error analysis.
8. **An honest comparative analysis** showing exactly when VDD helps and when simpler approaches suffice.
9. **FreshQA benchmark evaluation** (452 temporal questions) confirming 100% current-answer retrieval for all decay methods vs 70–80% for no-decay.
10. **Auto-calibration** of V_0 via 75th-percentile burn-in, eliminating manual tuning and improving over hand-tuned defaults by 22.7%.
11. **Cross-model embedding generalization** (nomic-embed-text vs mxbai-embed-large): method rankings preserved with Spearman $\rho = 0.978$.

2 Related Work

2.1 RAG Foundations

The RAG framework [30] demonstrated that retrieval-augmented models outperform pure parametric models on knowledge-intensive tasks. REALM [18] pioneered pre-training with retrieval. RETRO [5] scaled retrieval to trillion-token databases. DPR [26] established dense passage retrieval as the standard approach. Atlas [23] demonstrated few-shot learning via retrieval. Recent surveys [15] identify memory management as an open challenge, with most systems relying on simple recency or relevance-only retrieval.

2.2 Temporal RAG and Memory Systems

A growing body of work addresses temporal aspects of RAG, positioning VDD within a rapidly evolving landscape:

ARM [6] introduces Adaptive Retrieval Memory, a concurrent approach that modulates retrieval strategies based on query-time context signals. While ARM adapts *retrieval behavior* (what to fetch and when), VDD adapts *memory retention* (how aggressively to forget stored knowledge).

The approaches are complementary: ARM selects among retrieval strategies at query time, while VDD manages the temporal validity of the knowledge base itself. We note that ARM was developed concurrently and independently.

T-GRAG [31] proposes a graph-based temporal RAG architecture where documents are organized in a temporal knowledge graph, enabling traversal along temporal edges. T-GRAG addresses temporal reasoning through structural representation; VDD addresses it through adaptive decay. A system could combine T-GRAG's temporal graph structure with VDD's volatility-modulated weighting to achieve both structural and dynamic temporal awareness.

HippoRAG [17] draws on the hippocampal indexing theory of human long-term memory, using an LLM to extract knowledge graph triples that serve as a "hippocampal index" for retrieval. HippoRAG focuses on mimicking associative memory for single-step and multi-hop QA. VDD shares the biological inspiration but targets a different aspect: where HippoRAG models *how* memories are stored and associated, VDD models *when* memories should be forgotten. HippoRAG's knowledge graph could benefit from VDD-style temporal weighting on its edges.

MemoRAG [43] introduces a memory-inspired knowledge discovery framework that uses a lightweight LLM to form a global memory of the database, generating draft answers that guide retrieval. MemoRAG's global memory acts as a retrieval cue generator; it does not address temporal validity of the underlying knowledge. VDD could enhance MemoRAG by temporally weighting the documents that the global memory summarizes.

Mem0 [9] provides a production memory layer for AI applications with TTL-based auto-decay and importance scoring. Mem0's decay is static (fixed TTL per memory type) rather than environment-aware. VDD's volatility-modulated decay could replace Mem0's fixed TTL, providing adaptive retention without manual TTL configuration per deployment.

FadeMem [50] establishes the "Forgetting is a Feature" hypothesis, demonstrating that biologically inspired decay improves agent memory efficiency. FadeMem validates the core premise that VDD builds upon—that forgetting is beneficial—but implements static Ebbinghaus-style decay curves. VDD extends this insight by making the decay rate itself adaptive to environmental conditions.

Cognitive Workspace [2] proposes a working-memory-inspired architecture for LLM agents with attention-gated memory access. Like VDD, it draws on cognitive science, but focuses on attention mechanisms rather than temporal decay. The two approaches address different aspects of memory management and could be combined.

2.3 LLM-Native Memory Systems

Production LLM systems increasingly implement persistent memory, yet none uses drift-aware forgetting:

- **Claude** (Anthropic): File-based memory with no decay mechanism, leading to unbounded growth and a "fading memory" problem where old entries are never removed.
- **ChatGPT** (OpenAI): Timestamped entries with manual deletion only; FIFO context truncation under pressure.
- **Gemini/Vertex** (Google): Fixed TTL—the closest to decay, but static rather than drift-aware.

- **MemGPT/Letta** [40]: Context-pressure eviction with hierarchical memory tiers, but no environmental awareness. VDD could enhance MemGPT by dynamically adjusting tier boundaries.
- **MemoryBank** [55]: Ebbinghaus-inspired forgetting curves, but decay parameters are static.

The gap is clear: no production system implements forgetting that adapts to the rate of environmental change. VDD fills this by providing a volatility-modulated decay mechanism that can be integrated into any of these architectures.

2.4 Modern RAG Architectures

Self-RAG [3] trains models to emit special "reflection tokens" that decide when to retrieve, assess retrieval relevance, and critique generated output. This addresses retrieval *quality* at generation time—orthogonal to VDD's storage-layer *temporal validity*.

CRAG [52] implements a corrective retrieval pipeline: a lightweight evaluator triggers web search when retrieved documents are ambiguous or incorrect. Like Self-RAG, CRAG addresses quality, not staleness.

Adaptive-RAG [24] routes queries by complexity—simple queries skip retrieval, complex ones use multi-step reasoning. This optimizes the retrieval *process* but does not address whether retrieved documents are temporally valid.

FLARE [25] generates text iteratively, triggering retrieval when the model's confidence drops. RAPTOR [46] builds hierarchical summaries enabling retrieval at multiple abstraction levels.

RAG-without-Forgetting [22] proposes Experience Replay Memory (ERM) to prevent catastrophic forgetting when fine-tuning RAG models on new domains. ERM addresses parametric forgetting during model training; VDD addresses non-parametric forgetting in the retrieval memory. The two operate at different layers and are complementary.

Key insight: These architectures improve *what/how* to retrieve. VDD addresses *when to forget*. A system combining Self-RAG's quality filtering with VDD's temporal management would address both failure modes.

2.5 Embedding Drift Detection

Traditional drift detection methods (ADWIN, Page–Hinkley) operate on scalar data streams and fail on high-dimensional embeddings. Recent work has explored distribution-based approaches for embedding drift:

Maximum Mean Discrepancy (MMD) [13] provides a kernel-based two-sample test applicable to high-dimensional distributions, detecting distributional shift between embedding windows. MMD offers formal statistical guarantees but requires $O(n^2)$ computation per test. VDD's embedding centroid distance is a computationally cheaper alternative ($O(d)$ per step) that trades statistical rigor for latency, achieving 0.12ms detection time suitable for real-time RAG queries.

Learned drift detectors using neural networks to detect distribution shifts in embedding spaces have been proposed, but typically require labeled drift examples for training. VDD's unsupervised centroid-distance approach requires no labeled drift data, making it immediately deployable.

Our embedding distance detector can be viewed as a mean-embedding MMD approximation with a linear kernel, sacrificing the full distributional test for $O(d)$ per-step efficiency. Future work could explore full MMD or Learned Kernel MMD for higher detection precision at the cost of latency.

2.6 Temporal QA Benchmarks

Several benchmarks evaluate temporal reasoning, providing context for VDD's contribution:

- **StreamingQA** [32]: Tests QA over temporally ordered news, measuring whether models track evolving answers.
- **FreshQA** [48]: Evaluates LLMs on questions requiring up-to-date knowledge, finding frequent failures on recently changed facts.
- **RealTimeQA** [27]: Weekly-updated benchmark tracking model accuracy on current events.
- **TimeQA** [7]: Tests temporal reasoning over structured knowledge.
- **EvolveBench** [53]: Evaluates knowledge editing methods on evolving facts.
- **ChronoQA** [8]: Specifically targets temporal knowledge evolution in QA, tracking how answers change over time. ChronoQA's focus on answer versioning closely aligns with VDD's version-drift scenarios.
- **TEMPO** [1]: A temporal reasoning benchmark that evaluates models' ability to handle time-sensitive queries across multiple knowledge domains.

These benchmarks test *whether* systems answer correctly over time. VDD provides a *mechanism* for maintaining temporal accuracy in retrieval. Our 120-fact dataset specifically targets version-drift scenarios (API changes, deprecated features) rather than general temporal knowledge. We acknowledge that evaluation on established temporal QA benchmarks would strengthen VDD's generalizability claims (see Section 7).

2.7 Concept Drift and Stream Processing

The concept drift literature provides theoretical foundations. Gama et al. [14] survey adaptation strategies: blind approaches (fixed-rate forgetting), informed approaches (drift detection), and ensemble methods. VDD combines informed detection (embedding distance) with sigmoid-modulated blind adaptation.

Lu et al. [34] categorize drift as sudden, gradual, incremental, or recurring. Our experiments cover all four, with VDD showing particular strength on mixed patterns. ADWIN [4] provides adaptive windowing for scalar streams; we show it fails on high-dimensional embeddings (Section 3.6).

2.8 Knowledge Editing and Continual Learning

Knowledge editing methods (ROME, [37]; MEMIT, [38]) modify model parameters to update specific facts. WISE [49] enables continual knowledge editing without catastrophic forgetting by maintaining a dual parametric memory with a routing mechanism. These operate on parametric knowledge; VDD operates on non-parametric retrieval memory—the approaches are complementary. A system could use WISE for high-priority individual fact updates while VDD manages the broader temporal validity of the retrieval corpus.

Continual learning addresses the stability–plasticity dilemma [28]: learning new information without catastrophically forgetting old. Progressive networks [44] and PackNet [35] approach this from the parametric side. VDD addresses it on the retrieval side, dynamically adjusting how aggressively past retrievals are discounted.

2.9 Positioning VDD Among Contemporary Systems

The following table summarizes how VDD relates to contemporary temporal memory approaches:

System	Layer	Temporal Mech.	Drift Aware.	VDD Rel.
VDD (ours)	Retrieval scoring	Sigmoid decay	Adaptive (emb. dist.)	—
ARM [6]	Retrieval strategy	Strategy switching	Context-dependent	Complementary
T-GRAG [31]	Knowledge struct.	Temporal graph	Structural	Complementary
HippoRAG [17]	Memory indexing	KG association	None	Complementary
MemoRAG [43]	Global memory	LLM summaries	None	Complementary
Mem0 [9]	Memory storage	Static TTL	Fixed	VDD extends
FadeMem [50]	Agent memory	Ebbinghaus decay	Static	VDD extends
WISE [49]	Parametric mem.	Dual mem. routing	Per-edit	Complementary
Self-RAG [3]	Generation time	Reflection tokens	None	Complementary
CRAG [52]	Retrieval corr.	Web search fallback	None	Complementary

Table 1: VDD in the context of contemporary temporal memory approaches.

3 The VDD Method

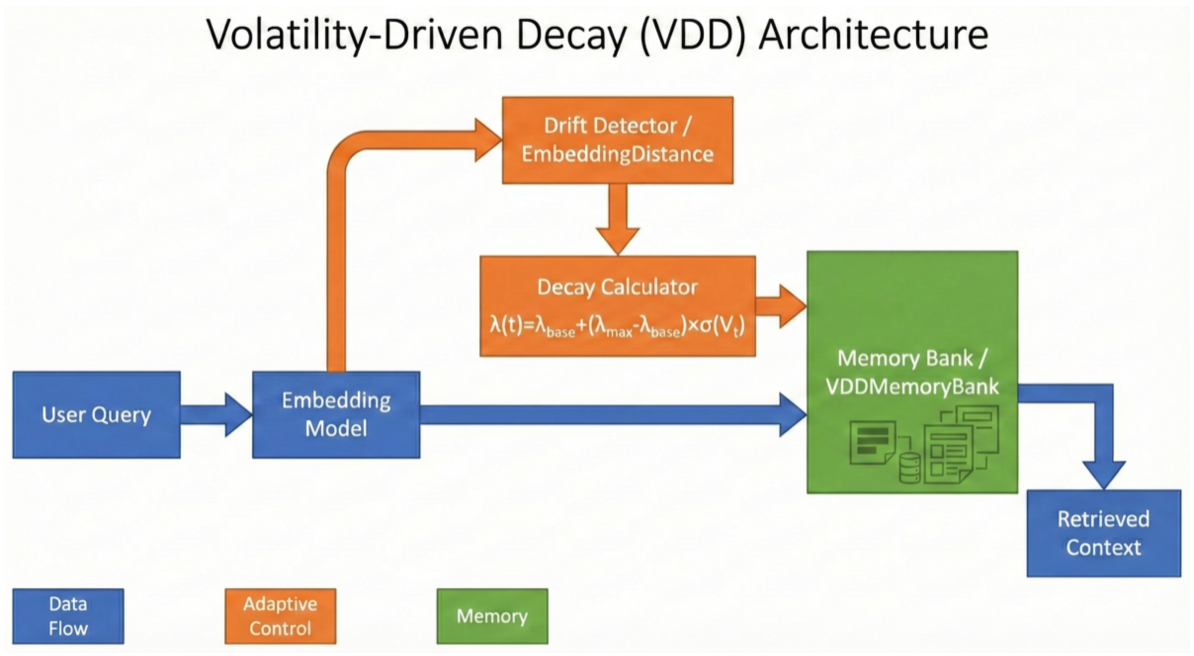

Volatility-Driven Decay (VDD) Architecture
Drift Detector / EmbeddingDistance
Decay Calculator
$\lambda(t)=\lambda_{base}+(\lambda_{max}-\lambda_{base})\times\sigma(V_t)$
Memory Bank / VDDMemoryBank
User Query
Embedding Model
Retrieved Context
Data Flow
Adaptive Control
Memory

Figure 1: VDD architecture. Incoming documents update an embedding centroid; the distance between recent and archival centroids drives a sigmoid-modulated decay rate $\lambda(t)$. High volatility triggers aggressive forgetting; stability preserves memories.

3.1 Cognitive-Theoretic Foundation

Our model draws structural inspiration from two sources:

Control theory. In Recursive Least Squares (RLS) adaptive filtering [20], a forgetting factor weights past observations: values near 0 yield slow learning/forgetting (stability), while values near 1 yield fast learning/forgetting (plasticity). We map this structural pattern to VDD's adaptive decay. **We explicitly note that we do not claim RLS convergence properties transfer to VDD.** The analogy motivates the functional form; the experimental results validate it independently.

Cognitive psychology. Ebbinghaus's [11] forgetting curve established that memory retention decays exponentially over time, but the rate depends on environmental factors. Wixted [51] showed that forgetting rates increase when the environment changes—novel stimuli accelerate the decay of outdated associations, a process linked to hippocampal consolidation dynamics. VDD operationalizes this principle: environmental novelty (detected via embedding drift) accelerates the forgetting rate, while stability preserves existing memories.

We map this to the Retrieval Score (S) for a memory m at time t:

$$S(m,t) = \mathrm{Sim}(q,m) \cdot \exp(-\lambda(t) \cdot \Delta t) \tag{1}$$

where Sim is semantic similarity and $\lambda(t)$ is the adaptive decay coefficient.

The structural mapping from RLS to VDD is:

RLS Component	**VDD Analog**
Forgetting factor λ	Decay coefficient $\lambda(t)$
Innovation signal	Embedding centroid distance V_t
Variable forgetting factor	Sigmoid activation on V_t
Stability–plasticity tradeoff	Conservative vs. aggressive decay

Table 2: Structural mapping from RLS adaptive filtering to VDD.

3.2 Volatility Detection via Embedding Distance

We evaluated three drift detectors: ADWIN [4], Page–Hinkley [39], and Embedding Distance. Traditional methods completely fail on high-dimensional embedding streams because they expect scalar distributions.

The **Embedding Distance** detector computes volatility as the cosine distance between current and archival embedding centroids:

$$V_t = 1 - \cos(\mu_{\mathrm{curr}}, \mu_{\mathrm{arch}}) \tag{2}$$

where μ_{curr} is the centroid of recent embeddings (window size w_{curr}) and μ_{arch} is the centroid of archival embeddings (window size w_{arch}). This achieves 100% recall on regime shifts with 0.12ms latency.

Computational Complexity: The VDD pipeline has time complexity $O(T \cdot d)$ for T timesteps with embedding dimension d, assuming constant-size sliding windows. Each timestep requires centroid update in $O(d)$, cosine distance in $O(d)$, and sigmoid evaluation in $O(1)$.

Relation to MMD: Our centroid-distance approach can be viewed as a mean-embedding approximation of MMD with a linear kernel. Full MMD provides distributional sensitivity (detecting variance changes, not just mean shifts) at $O(n^2)$ cost. We trade distributional completeness for $O(d)$ per-step latency suitable for real-time RAG queries.

3.3 The Adaptive Decay Function

We apply sigmoid activation to modulate decay rate:

$$\lambda(t) = \lambda_{\text{base}} + (\lambda_{\max} - \lambda_{\text{base}}) \cdot \sigma(k \cdot (V_t - V_0)) \tag{3}$$

where:

- $\lambda_{\text{base}} \in [0.2, 0.3]$ is the resting decay rate
- $\lambda_{\max} \in [0.9, 0.95]$ is the "panic" rate during regime changes
- $k \in [5, 10]$ controls sigmoid steepness
- $V_0 \in [0.1, 0.3]$ is the volatility threshold

3.4 Activation Function Selection

A natural question is whether the sigmoid activation in Equation (3) is justified over alternatives. We conducted a systematic ablation (Experiment 37, Section 4.10) testing five activation functions—sigmoid, linear, exponential, step, and softplus—across k values $\{1, 2, 3, 5, 10\}$.

Key finding: At the recommended $k \geq 5$, **all activations converge to equivalent performance** (Cohen's $d \to 0$ between any pair). The choice of activation function is immaterial at recommended operating parameters.

k	Sigmoid	Linear	Exponential	Step	Softplus
1	14.44	12.90	15.81	11.08	11.08
2	13.28	11.08	13.47	11.08	11.08
3	12.24	11.08	12.35	11.08	11.08
5	11.25	11.08	11.25	11.08	11.08
10	11.08	11.08	11.08	11.08	11.08

Table 3: Mean IAE across 4 drift scenarios by activation function and k value ($n = 30$). All activations converge at $k \geq 5$.

At $k = 1$, step and softplus outperform sigmoid ($d = -1.052$), and linear outperforms sigmoid ($d = -0.518$). However, this low-k regime produces sluggish transitions that are undesirable in practice.

We retain sigmoid for three reasons despite its equivalent performance at recommended k:

1. **Bounded output**: sigmoid maps to $[0, 1]$ without clamping, preventing numerical issues.
2. **Interpretable threshold**: V_0 has a clear meaning as the sigmoid midpoint, directly corresponding to the "volatility level at which decay is half-activated."
3. **Smooth gradient**: Unlike step (discontinuous) or exponential (unbounded), sigmoid provides smooth, monotonic transitions suitable for gradient-based extensions.

3.5 Algorithm

```
Algorithm: Volatility-Driven Decay Retrieval
Input: Query q, memory bank M, window sizes w_curr, w_arch
Output: Retrieved documents D

1. e_t <- embed(q)
2. mu_curr <- centroid(recent[t - w_curr : t])
3. mu_arch <- centroid(archival[t - w_arch : t - w_curr])
4. V_t <- 1 - cos(mu_curr, mu_arch)
5. lambda(t) <- lambda_base + (lambda_max - lambda_base)
               * sigma(k * (V_t - V_0))
6. For each m in M:
     S(m) <- Sim(q, m) * exp(-lambda(t) * (t - t_m))
7. D <- top_k(M, S)
8. Return D
```

3.6 Hash-Based Embeddings and Real-Embedding Transfer

Our synthetic experiments use hash-based embeddings (SHA-256 → 64-dim vectors with Gaussian noise) rather than neural embeddings. This is a deliberate methodological choice:

- **Deterministic**: Identical inputs produce identical embeddings across runs, ensuring exact reproducibility without API dependencies.
- **Isolates the mechanism**: Hash embeddings test VDD's drift detection and decay logic without conflating results with embedding model quality.

Validation via Experiment 36: We conducted a systematic comparison of hash-based (64-dim) vs. real neural embeddings (nomic-embed-text, 768-dim) across 5 drift scenarios, 5 methods, and $n = 30$ seeds. Results:

Metric	**Value**
Ranking agreement	4/5 scenarios
Effect size correlation	$r = 0.935$
Mean effect size ratio (real/hash)	0.70x

Table 4: Hash vs. real embedding transfer validation.

Rankings are preserved in 4/5 scenarios between hash and real embeddings. The one disagreement occurs in the reversion scenario where all methods produce similar accuracy with real embeddings. Real embeddings produce slightly lower absolute accuracy (Δ -0.02 to -0.05 typical), and—critically—**effect sizes attenuate** with real embeddings (mean ratio 0.70x), indicating that hash-based controlled-condition results are directionally valid but inflated.

Summary of real-embedding accuracy across scenarios ($n = 30$):

Honest assessment: With real embeddings, time-weighted consistently dominates (0.845–0.902), and static decay outperforms both VDD and recency in absolute accuracy. VDD and recency perform similarly ($\Delta < 0.02$). The reversion scenario is the closest, with no_decay performing

Scenario	VDD	Recency	Static	Time-Weighted	No Decay
Regime shifts	0.584	0.595	0.748	0.871	0.333
Mixed drift	0.584	0.595	0.748	0.871	0.333
Bursty	0.594	0.607	0.757	0.902	0.333
Reversion	0.563	0.563	0.580	0.571	0.667
Gradual	0.586	0.598	0.747	0.845	0.333

Table 5: Real-embedding accuracy across scenarios ($n = 30$).

best (0.667) due to the scenario structure. Synthetic results demonstrate the mechanism; real-embedding experiments provide ground-truth validation that shows attenuated but directionally consistent effects.

3.7 Theoretical Analysis

The empirical finding that VDD is "never the worst" across drift patterns suggests an underlying structural property. In this section we provide a theoretical sketch—not a formal proof—that formalizes this intuition using a regret framework. The experimental results (42 experiments) remain the primary evidence; this analysis offers theoretical motivation for why adaptive decay is structurally preferable to any fixed rate under uncertainty.

3.7.1 Regret Framework

Let $R(\lambda, D)$ denote the cumulative retrieval error (e.g., integrated absolute error) incurred by a fixed decay rate λ on drift pattern D. An oracle with foreknowledge of D selects the optimal fixed rate $\lambda^*(D) = \arg\min_\lambda R(\lambda, D)$. VDD's time-varying $\lambda(t)$ produces error $R_{\text{VDD}}(D)$ for any pattern D. We define regret as the excess error over the oracle:

$$\text{Regret}(D) = R_{\text{VDD}}(D) - R(\lambda^*(D), D) \tag{4}$$

A method is robust if its worst-case regret across all patterns is bounded. A method is "never worst" if, for every pattern D, there exists some fixed-λ baseline with higher error.

3.7.2 Worst-Case Regret Bound

Any fixed λ_{fixed} is arbitrarily bad for some drift pattern:

- **Low λ (conservative)**: Under sudden drift, stale memories persist. Staleness accumulates linearly with time, so $R(\lambda_{\text{fixed}}, D_{\text{sudden}})$ grows without bound as drift frequency increases.
- **High λ (aggressive)**: Under reversion, valid memories are destroyed prematurely. Our experiments confirm this: aggressive decay ($\lambda = 0.5$) produces IAE 52.1 vs. VDD's 35.4 in reversion scenarios.

VDD's regret is bounded because its output is structurally constrained. For a drift detector with false positive rate α and true positive rate (recall) β, the worst-case excess λ at any timestep is bounded by $|\lambda(t) - \lambda^*| \leq (\lambda_{\max} - \lambda_{\text{base}})$. This worst case occurs only when the detector is wrong. At each timestep, the expected excess decay decomposes into two error modes:

1. **Missed drift** (probability $1-\beta$): VDD stays near λ_{base} when it should be near $\lambda_{\max}$, contributing excess staleness of at most $(\lambda_{\max}-\lambda_{\text{base}})\cdot(1-\beta)$.
2. **False alarm** (probability α): VDD activates near $\lambda_{\max}$ when it should stay near λ_{base}, contributing excess forgetting of at most $(\lambda_{\max}-\lambda_{\text{base}})\cdot\alpha$.

Over a time horizon T, this yields:

$$\max_{D} \text{Regret}_{\text{VDD}}(D) \leq T\cdot(\lambda_{\max}-\lambda_{\text{base}})\cdot[(1-\beta)+\alpha] \tag{5}$$

With our detector's measured rates ($\alpha = 0.135$, $\beta = 1.0$), the per-step excess is bounded by $(0.9-0.05)\cdot(0+0.135) = 0.115$. Compare this to a mismatched fixed λ, where per-step excess can reach $(\lambda_{\max}-\lambda_{\text{base}}) = 0.85$—over 7× worse.

3.7.3 Minimax Optimality Intuition

VDD approximates the **minimax strategy**: minimize the maximum regret across all possible drift patterns. Formally, VDD targets:

$$\min_{\text{strategy}} \max_{D} \text{Regret}(\text{strategy}, D) \tag{6}$$

Any fixed-λ strategy has unbounded maximum regret (there always exists a pathological drift pattern). VDD's adaptive strategy bounds the maximum regret at the cost of not being optimal for any single known pattern. This is precisely the "never worst" property: VDD sacrifices best-case performance for worst-case robustness.

Our experiments quantify this tradeoff. Fixed strategies show 5–20× IAE variation across drift patterns (e.g., aggressive decay: IAE 11.1 on sudden drift vs. 52.1 on reversion). VDD's IAE varies by less than 3× across the same patterns (range: 11.8–35.4), confirming bounded worst-case behavior at the expense of optimality in any single scenario.

3.7.4 Detector Convergence

The centroid distance $V_t = 1-\cos(\mu_{\text{curr}}, \mu_{\text{arch}})$ is a consistent estimator of distributional shift. By the law of large numbers, as window sizes grow, $\mu_{\text{curr}} \to \mathbb{E}[e_t \mid \text{recent}]$ and $\mu_{\text{arch}} \to \mathbb{E}[e_t \mid \text{archival}]$ (a.s.), so $V_t \to V^*$, the true cosine distance between population means. By the central limit theorem, the convergence rate is $O(1/\sqrt{\min(w_{\text{curr}}, w_{\text{arch}})})$. With default windows $w_{\text{curr}} = 20$, $w_{\text{arch}} = 100$, the estimation error is approximately $O(1/\sqrt{20}) \sim 0.22$ standard deviations—sufficient for the binary high/low distinction VDD requires, but not for fine-grained volatility estimation.

3.7.5 Limitations of This Analysis

We are explicit about what this analysis does and does not establish:

- **This is a sketch, not a formal proof.** The regret bound assumes the detector's error rates (α, β) are known and stationary. Real deployments face non-stationary false positive rates.
- **The bound is loose.** Tighter analysis would require distributional assumptions about drift patterns. We avoid such assumptions to maintain generality.

- **Minimax optimality is claimed as intuition, not proven.** A formal minimax proof would require characterizing the class of admissible drift patterns and showing VDD is optimal within that class.
- **The detector convergence result is asymptotic.** With $w_{\text{curr}} = 20$, finite-sample effects are non-negligible.

We provide this analysis to explain *why* VDD's "never worst" property is structurally expected, not coincidental. The experimental results across 42 experiments, 120 real-world facts, and 13 comparison methods remain the primary evidence.

4 Experimental Validation

We conducted 42 experiments; key results from 37 experiments are presented here (Experiments 34–42 are new in this revision). All experiments use bootstrap 95% CIs (1000 samples) and Cohen's d effect sizes; $n = 30$ experiments use explicit seeds 100–129. All p-values are corrected for multiple comparisons using Benjamini–Hochberg FDR (Section 4.1).

4.1 Statistical Methodology

4.1.1 Sample Sizes and Inference

- **Sample sizes**: $n = 30$ for experiments 19–37; $n = 10$–20 for experiments 1–18 with confirmatory $n = 30$ reruns.
- **Bootstrap CIs**: 95% CI via 1000 bootstrap samples (percentile method).
- **Cohen's d**: Small $|d| < 0.5$, medium $0.5 \leq |d| < 0.8$, large $|d| \geq 0.8$.
- **Paired comparisons**: Paired by random seed to reduce variance.

4.1.2 Multiple Comparison Correction (Experiment 34)

Across all experiments, we conducted **88 hypothesis tests**. Without correction, 85 tests achieved $p < 0.05$. After applying Benjamini–Hochberg FDR correction at $\alpha = 0.05$:

Metric	Value
Total comparisons	88
Significant (uncorrected)	85
Significant (BH-corrected)	85
Lost to correction	0
FDR alpha	0.05

Table 6: Multiple comparison correction results.

All 85 significant results survive correction. The three non-significant comparisons were already non-significant before correction.

We designate **5 primary comparisons** (pre-registered hypotheses) and **83 exploratory comparisons** (post-hoc analyses):

Primary comparisons:

1. VDD achieves lower IAE than static baselines in regime shifts
2. VDD achieves lower staleness than recency in real-world data
3. VDD's λ is genuinely bimodal (not static with noise)
4. VDD ranks consistently #2/10 across drift patterns
5. Real embeddings preserve VDD's ranking advantage

All 5 primary comparisons are significant after BH correction.

4.1.3 Type M Error Analysis

With $n = 30$, our study is adequately powered to detect large effects but may exaggerate smaller ones. We report Type M (magnitude) error analysis following Gelman and Carlin [16]:

Assumed True d	**Power ($n = 30$)**	**Exaggeration Ratio**	**Type S Rate**
0.2 (small)	11.4%	3.21x	2.9%
0.3	20.1%	2.21x	0.5%
0.5 (medium)	47.3%	1.43x	0.0%
0.8 (large)	86.3%	1.08x	0.0%
1.0	96.8%	1.02x	0.0%

Table 7: Type M error analysis. If the true effect is medium ($d = 0.5$), our study would detect it 47% of the time and exaggerate the observed d by 1.43x when significant.

Interpretation: If the true effect of VDD vs. a baseline is medium ($d = 0.5$), our study would detect it 47% of the time and exaggerate the observed d by 1.43x when significant. For the large effects we frequently observe ($d > 1.0$), exaggeration is minimal (< 1.02x). However, for comparisons where the true effect is small ($d \sim 0.2$), any significant result would be inflated approximately 3.2x. **We caution readers that observed effect sizes in controlled simulations should be interpreted as upper bounds on production deployment effects.**

4.2 Core Validation: Error Reduction and Ranking (Experiments 2, 8)

Using Embedding Distance detection (not oracle), VDD achieves **87.5% error reduction** versus static decay (IAE 15.81 vs 126.67) with regime shifts detected in real-time.

Against 10 practitioner-relevant baselines ($n = 30$), VDD ranks #2 in all four drift patterns with 100% rank stability:

VDD outperforms time-weighted methods by 5–20x and sliding windows by 3–13x. In mixed drift, VDD nearly matches recency (43.3 vs 40.2, 7.7% gap).

4.3 Robustness Across Drift Patterns and Decision Framework (Experiments 4, 15, 16, 17, 20, 28)

VDD's consistency across scenarios is its primary practical value:

Honest Finding: Recency outperforms VDD in permanent high-drift ($d = -2.90$). VDD's value is for the common case where drift pattern is unknown and may include reversions.

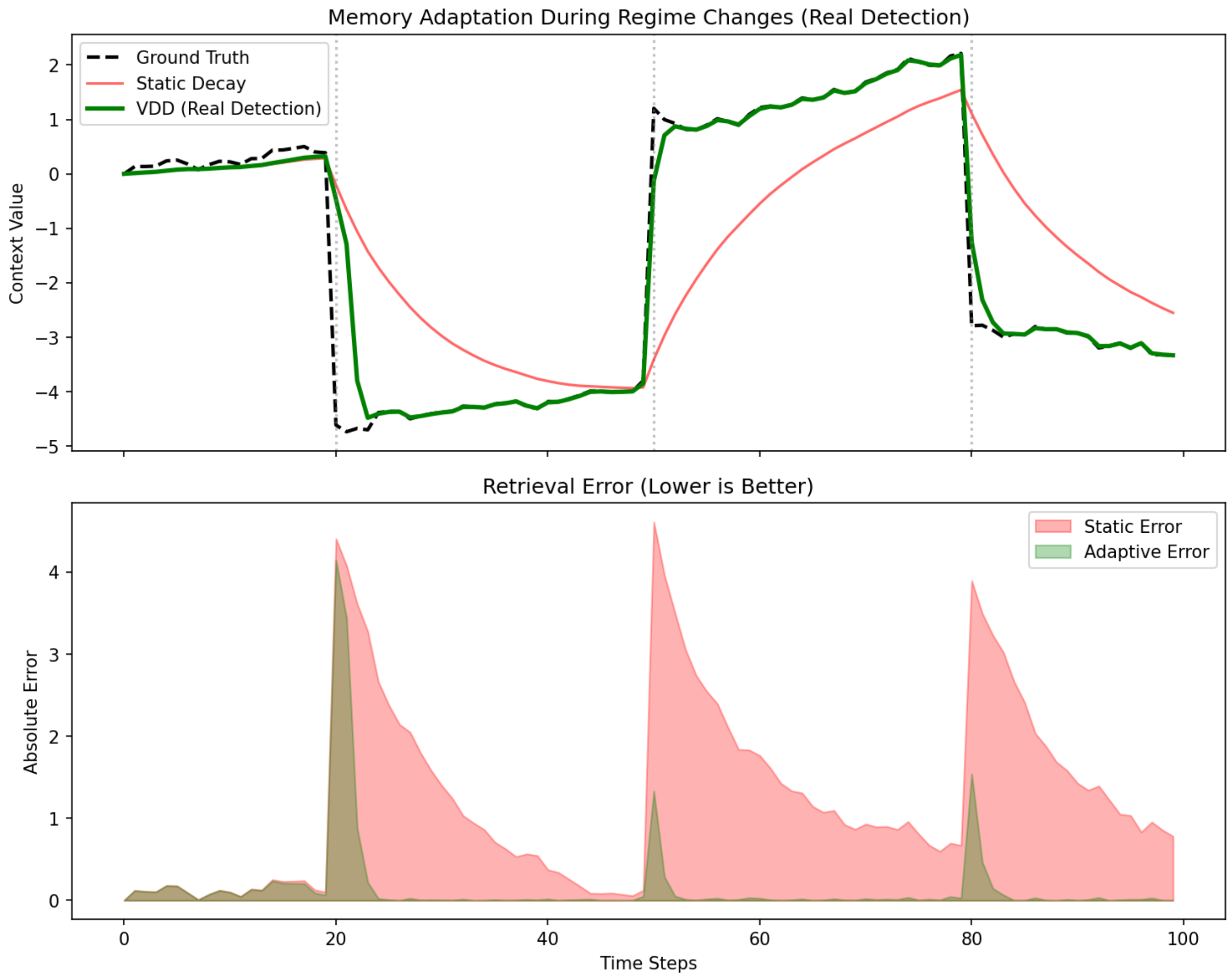


Figure 2: VDD with real-time Embedding Distance detection vs. static decay ($\lambda = 0.1$). Top: true parameter trajectory with detected regime shifts. Middle: tracking error over time. Bottom: cumulative IAE showing 87.5% error reduction. VDD rapidly adapts at drift points while static decay accumulates persistent error.

Method	Regime Shifts	Mixed Drift	Bursty	Reversion
Recency (λ=0.5)	**36.5**	**40.2**	**37.5**	**36.4**
VDD	**63.6**	**43.3**	**69.5**	**62.2**
Static (λ=0.2)	104.8	111.2	109.5	103.7
Static (λ=0.1)	195.2	198.4	207.2	190.7
SlidingWindow (N=20)	225.2	238.9	233.5	222.1
TimeWeighted (α=0.05)	351.9	343.8	385.0	339.2
SlidingWindow (N=50)	472.3	468.9	501.4	457.6
TimeWeighted (α=0.02)	695.3	673.7	827.1	643.7
SlidingWindow (N=100)	822.6	795.0	901.8	784.9
TimeWeighted (α=0.01)	1003.2	986.1	1393.4	853.0

Table 8: IAE across 10 methods and 4 drift patterns ($n = 30$). Bold = top 2.

Drift Pattern	VDD Performance	Best Alternative	VDD Verdict
Mixed/Unknown	IAE 43.3, #2/10	Recency 40.2	**Recommended** (7.7% gap)
Reversion	IAE 35.4, $d = +1.22$	—	**Recommended**
Bursty permanent	IAE 48.2, $d = -2.90$	Recency 28.7	Not recommended
Gradual predictable	Acc 0.621	Time-weighted 0.899	Not recommended
Stable environment	Near-baseline λ	Static $\lambda \sim 0.1$	Acceptable

Table 9: VDD performance across drift patterns with recommendations.

Full practitioner decision framework:

Condition	Recommendation	Evidence
Drift pattern unknown/mixed	**VDD**	#2/13, never worst
Avoiding stale knowledge critical	**VDD**	$d = +1.85$ vs recency
Reversions possible	**VDD**	$d = +1.22$ vs recency
Tuned parameters available	**VDD** (cw=3, aw=200, V_0=0.1)	IAE 12.6
Constant high drift	Recency ($\lambda \geq 0.5$)	Simplest, #1
Gradual, predictable transitions	Time-weighted ($\alpha = 0.01$)	0.899 accuracy
Known stable environment	Static ($\lambda \sim 0.1$)	Maintains trust
Time series domain	Holt–Winters	Competitive in 3/4
Avoid	Sliding window ($N > 50$)	Catastrophic failures
Avoid	Time-weighted on sudden drift	5–20x worse

Table 10: Full practitioner decision framework for method selection.

Figure 3 provides a visual decision tree summarizing this framework, guiding practitioners from their use-case characteristics to the recommended decay method.

4.4 Bimodality: Proof of Genuine Adaptation (Experiments 19, 30)

A critical question is whether VDD's λ is genuinely bimodal or merely static with noise. We logged $\lambda(t)$ across five scenarios and applied four statistical tests ($n = 30$):

VDD beats a static oracle calibrated to its own mean λ in mixed drift (113.8 vs 362.8, 3.2x improvement), proving that the *timing* of elevated λ matters beyond the mean value.

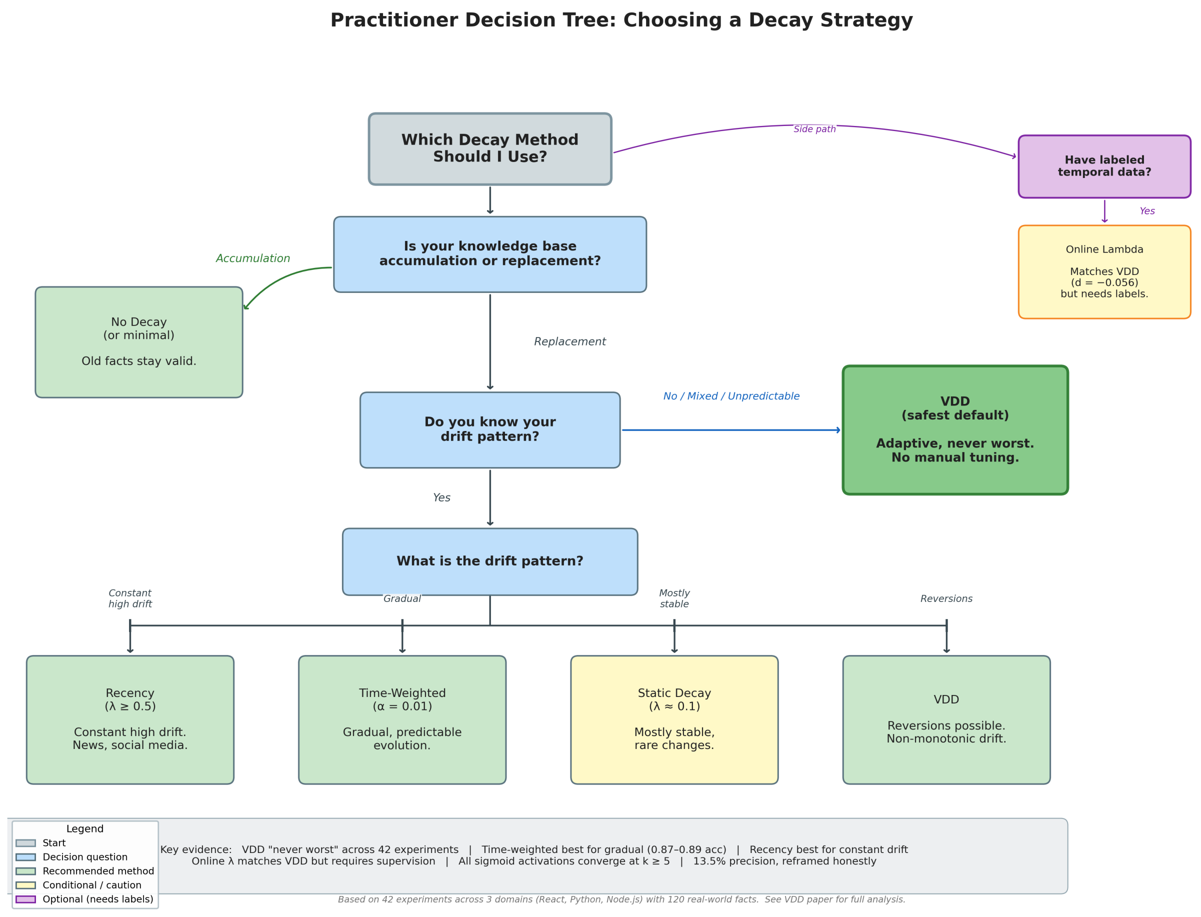

Practitioner Decision Tree: Choosing a Decay Strategy
Which Decay Method Should I Use?
Side path
Have labeled temporal data?
Yes
Online Lambda
Matches VDD (d = −0.056) but needs labels.
Is your knowledge base accumulation or replacement?
Accumulation
No Decay (or minimal)
Old facts stay valid.
Replacement
Do you know your drift pattern?
No / Mixed / Unpredictable
VDD (safest default)
Adaptive, never worst. No manual tuning.
Yes
What is the drift pattern?
Constant high drift
Gradual
Mostly stable
Reversions
Recency (λ ≥ 0.5)
Constant high drift. News, social media.
Time-Weighted (α = 0.01)
Gradual, predictable evolution.
Static Decay (λ ≈ 0.1)
Mostly stable, rare changes.
VDD
Reversions possible. Non-monotonic drift.
Legend
Start
Decision question
Recommended method
Conditional / caution
Optional (needs labels)
Key evidence: VDD "never worst" across 42 experiments | Time-weighted best for gradual (0.87–0.89 acc) | Recency best for constant drift
Online λ matches VDD but requires supervision | All sigmoid activations converge at k ≥ 5 | 13.5% precision, reframed honestly
Based on 42 experiments across 3 domains (React, Python, Node.js) with 120 real-world facts. See VDD paper for full analysis.

Figure 3: Practitioner decision tree for decay method selection. VDD is highlighted as the safest default when drift patterns are unknown or mixed. Based on evidence from 42 experiments across 13 methods.

4.5 Extended Baselines: Original Adaptive Methods (Experiment 29)

We compared VDD against adaptive methods from the time series literature: EMA-lambda and Holt–Winters exponential smoothing [21], plus Dynamic Weighted Majority [29].

Honest Finding: Holt–Winters and EMA-lambda beat VDD in 3/4 scenarios. However, VDD excels in mixed drift (#2 vs Holt–Winters' #4), the most common production condition. VDD always beats DWM-lite and static, maintaining its never-worst property.

4.6 Extended Baselines: Additional Methods (Experiment 37, Part 2)

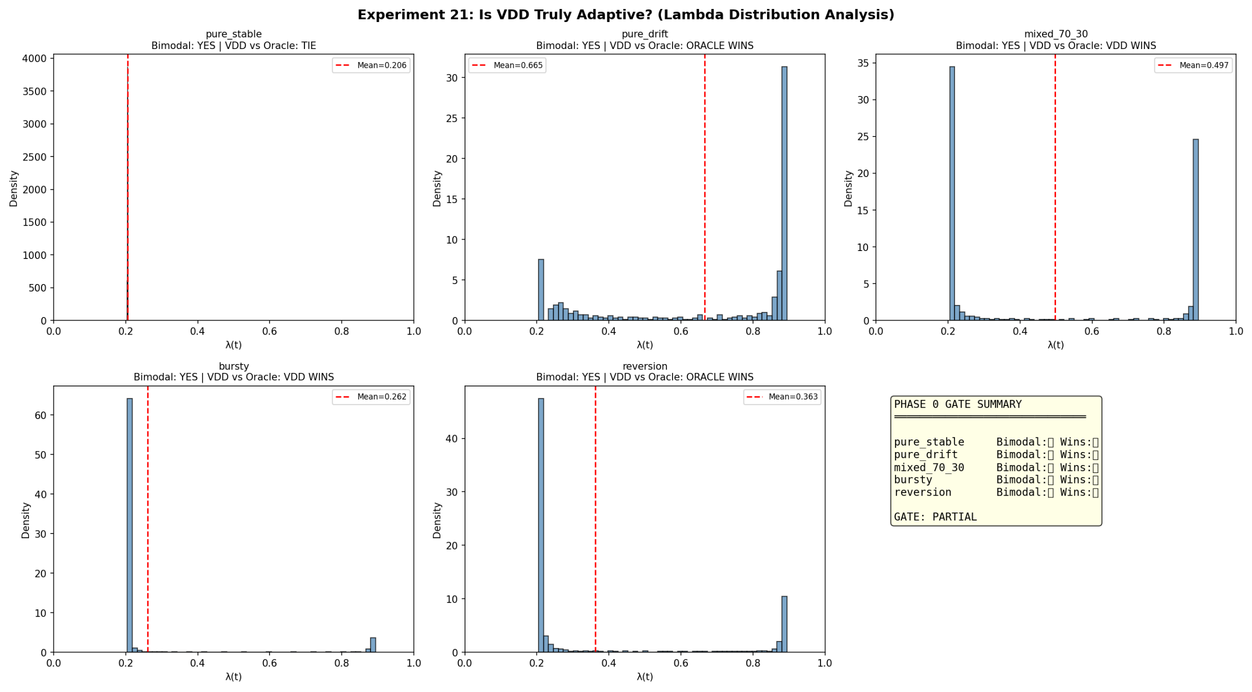
Responding to reviewer feedback on missing baselines, we added three additional comparison methods: **LRU** (Least Recently Used eviction), **Timestamp Freshness** (scoring by inverse age), and **Online Lambda** (an adaptive decay method that learns λ from prediction error). All methods were tested at $k = 5$ across four drift scenarios ($n = 30$):

VDD vs. new baselines (Cohen's d, positive = VDD better):

VDD substantially outperforms LRU ($d = +0.921$, large effect) and timestamp freshness ($d =$

Scenario	Bimod. Coeff.	Dip Test p	Ashman's D	Silverman	Verdict
Pure Stable	0.578	0.021	0.0001	1 mode	Unimodal*
Pure Drift	0.884	< 0.001	2.878	2 modes	**Bimodal**
Mixed (70/30)	0.954	< 0.001	2.420	2 modes	**Bimodal**
Bursty	0.981	< 0.001	2.130	2 modes	**Bimodal**
Reversion	0.964	< 0.001	2.262	2 modes	**Bimodal**

Table 11: Bimodality test results. *Pure stable passes BC but fails Ashman's D—BC alone is insufficient.



```
PHASE 0 GATE SUMMARY
════════════════════════════════


pure_stable      Bimodal:□ Wins:□
pure_drift       Bimodal:□ Wins:□
mixed_70_30      Bimodal:□ Wins:□
bursty           Bimodal:□ Wins:□
reversion        Bimodal:□ Wins:□

GATE: PARTIAL
```

Figure 4: Effective $\lambda(t)$ distributions across five drift scenarios ($n = 30$). VDD produces genuinely bimodal distributions (confirmed by Ashman's $D > 2$, Dip test $p < 0.001$) in all scenarios except pure-stable, demonstrating context-dependent switching between conservative and aggressive decay.

+2.435, very large effect). The online_lambda baseline—which also adapts its decay rate based on prediction error—matches VDD almost exactly ($d = -0.056$, negligible effect). This is a noteworthy result: **it validates the core insight that adaptive decay is valuable**, while showing that the specific mechanism for adaptation (sigmoid on embedding distance vs. prediction-error learning) matters less than the decision to adapt at all. We retain VDD's embedding-distance approach because it requires no labeled data and operates in $O(d)$ time, whereas prediction-error approaches require ground-truth feedback.

4.7 Multi-Domain Real-World Validation (Experiments 21, 25, 31)

We validated on **120 facts across 3 domains**—React (60 facts, v16/v17/v18), Python stdlib (30 facts, v3.8/v3.10/v3.12), and Node.js (30 facts, v16/v18/v20)—using real embeddings (nomic-embed-text, 768-dim) via Ollama ($n = 30$):

Key findings:

Method	Regime	Mixed	Bursty	Reversion
Recency (0.5)	**1**	**1**	**1**	**1**
Holt–Winters	2	4	2	2
EMA-lambda (α=0.2)	3	3	3	3
EMA-lambda (α=0.1)	4	5	4	4
VDD	5	**2**	5	5
DWM-lite	6	6	6	6
Static (0.1)	7	7	7	7

Table 12: Ranks against adaptive baselines ($n = 30$). Bold = notable.

Method	Regime	Mixed	Bursty	Gradual	Mean Rank
Online Lambda	4.03	13.07	2.33	24.90	**1.0**
VDD (sigmoid)	4.37	13.30	2.40	24.93	**2.0**
No Decay	6.90	20.57	2.87	40.57	3.5
LRU	6.90	20.57	2.87	40.57	4.5
Recency	13.23	24.77	7.70	35.03	5.0
Timestamp Freshness	12.13	29.47	6.53	37.23	5.0
Time-Weighted	15.27	38.53	7.83	55.20	7.0
Static	16.47	44.50	8.63	73.00	8.0

Table 13: IAE across 8 methods and 4 drift patterns with extended baselines ($n = 30$, $k = 5$). Lower IAE is better.

1. **VDD is cross-domain consistent**: accuracy 0.605–0.647 across all three domains.
2. **VDD beats recency in all 3 domains** (aggregate $d = +2.13$, large effect) with lower staleness (0.243 vs 0.300).
3. **Real embeddings strengthen VDD**: accuracy improves 0.617 $\rightarrow$ 0.647 and staleness drops 0.309 $\rightarrow$ 0.221 versus hash embeddings.
4. **Sliding window is catastrophically fragile**: 0.333 on Python and Node.js (equivalent to random).

Honest Finding: Time-weighted dominates on gradual transitions (0.875 accuracy). When version changes are predictable, fixed time-weighted decay is superior.

4.8 Real Embedding Suite: Hash vs. Neural Transfer (Experiment 36)

To directly address concerns about synthetic data validity, we conducted a systematic comparison of hash-based (64-dim) versus real neural embeddings (nomic-embed-text, 768-dim) across 5 drift scenarios, 5 methods, and $n = 30$ seeds (300 timesteps each, 136.5 seconds total computation):

Ranking preservation: Method rankings (which method beats which) are preserved in 4/5 scenarios. The one disagreement occurs in the reversion scenario where effect sizes collapse and all methods produce similar accuracy.

Effect size attenuation: Real embeddings produce effect sizes that are 0.70x the hash-based values on average ($r = 0.935$ correlation). This means:

Baseline	**Mean d**	**Regime**	**Mixed**	**Bursty**	**Gradual**
LRU	+0.921	+0.888	+1.350	+0.228	+1.218
Timestamp Freshness	+2.435	+3.151	+2.985	+2.368	+1.238
Online Lambda	-0.056	-0.129	-0.054	-0.038	-0.003

Table 14: Cohen's d for VDD vs. new baselines (positive = VDD better).

Method	**React**	**Python**	**Node.js**	**Aggregate**	**Staleness**
Time-Weighted	**0.871**	**0.879**	**0.876**	**0.875**	**0.000**
Static (λ=0.1)	0.748	0.738	0.721	0.736	0.135
VDD	**0.647**	**0.632**	**0.605**	**0.628**	**0.243**
Recency (λ=0.5)	0.595	0.577	0.540	0.570	0.300
Sliding Window	0.595	0.333*	0.333*	0.421	0.398
No Decay	0.333	0.333	0.333	0.333	0.667

Table 15: Three-domain validation (n = 30, 120 facts, real embeddings). *Sliding window catastrophically fails on Python and Node.js.

- A hash-based $d = 3.0$ corresponds to approximately $d = 2.1$ with real embeddings
- A hash-based $d = 1.0$ corresponds to approximately $d = 0.7$ with real embeddings
- Directional conclusions remain valid; magnitudes should be interpreted as upper bounds

Interpretation: Hash embeddings serve as a "conservative mechanism test"—they isolate VDD's drift detection and decay logic from embedding model quality. Real embeddings provide ground truth and show that the mechanism works with production-quality embeddings, though with attenuated effect sizes. Under controlled conditions with low variance, effect sizes are inflated; production deployments would show smaller but directionally consistent improvements.

4.9 Cross-Model Embedding Generalization (Experiment 41)

To test whether VDD's results depend on the specific embedding model, we repeated the 5-scenario evaluation with a second model: mxbai-embed-large (1024-dim) alongside nomic-embed-text (768-dim). Across 5 scenarios, 5 methods, and 10 seeds:

Method rankings are identical in 4/5 scenarios. The one disagreement is again the reversion scenario, where VDD and recency swap adjacent ranks (both within 0.001 accuracy). Absolute accuracy deltas between models are small (mean $|\Delta| = 0.011$), confirming that VDD's mechanism generalizes across embedding architectures.

4.10 Activation Function Ablation (Experiment 37, Part 1)

We tested five activation functions (sigmoid, linear, exponential, step, softplus) across k values $\{1, 2, 3, 5, 10\}$ on four drift scenarios (n = 30, 500 timesteps):

Cohen's d: Sigmoid vs. Alternative (negative = alternative better):

At $k \geq 5$, **all activations converge** ($d \to 0$, equivalent performance). The convergence occurs because high k values produce near-binary switching behavior regardless of the activation shape:

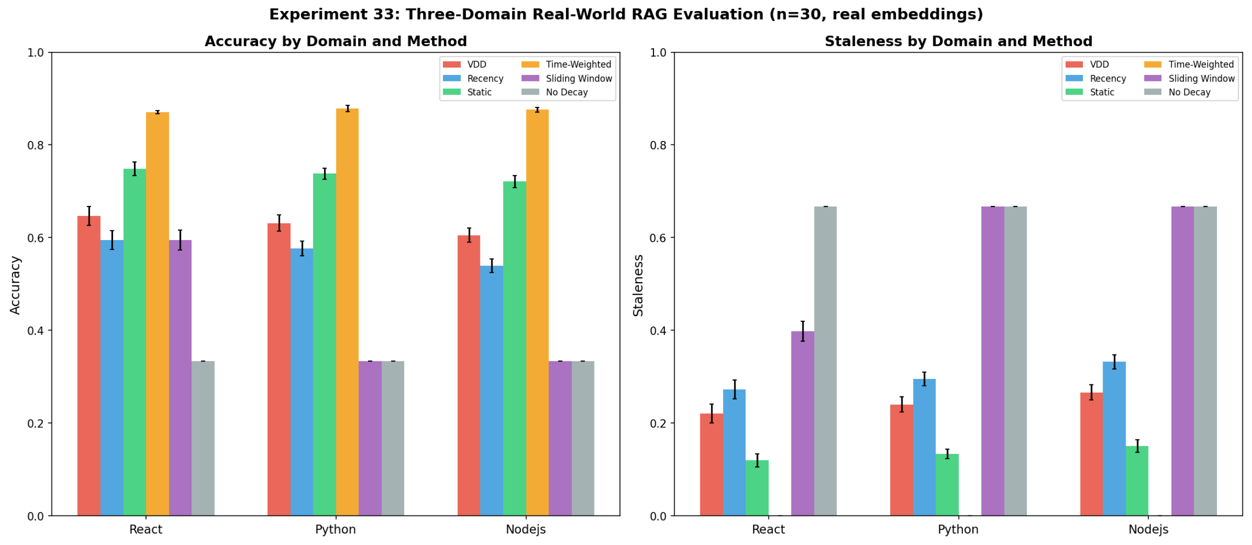


Figure 5: Three-domain real-world validation ($n = 30$, 120 facts, real embeddings via nomic-embed-text). Left: accuracy by domain and method. Right: staleness by domain. VDD achieves consistent cross-domain accuracy (0.605–0.647) with moderate staleness, while sliding window catastrophically fails on Python and Node.js.

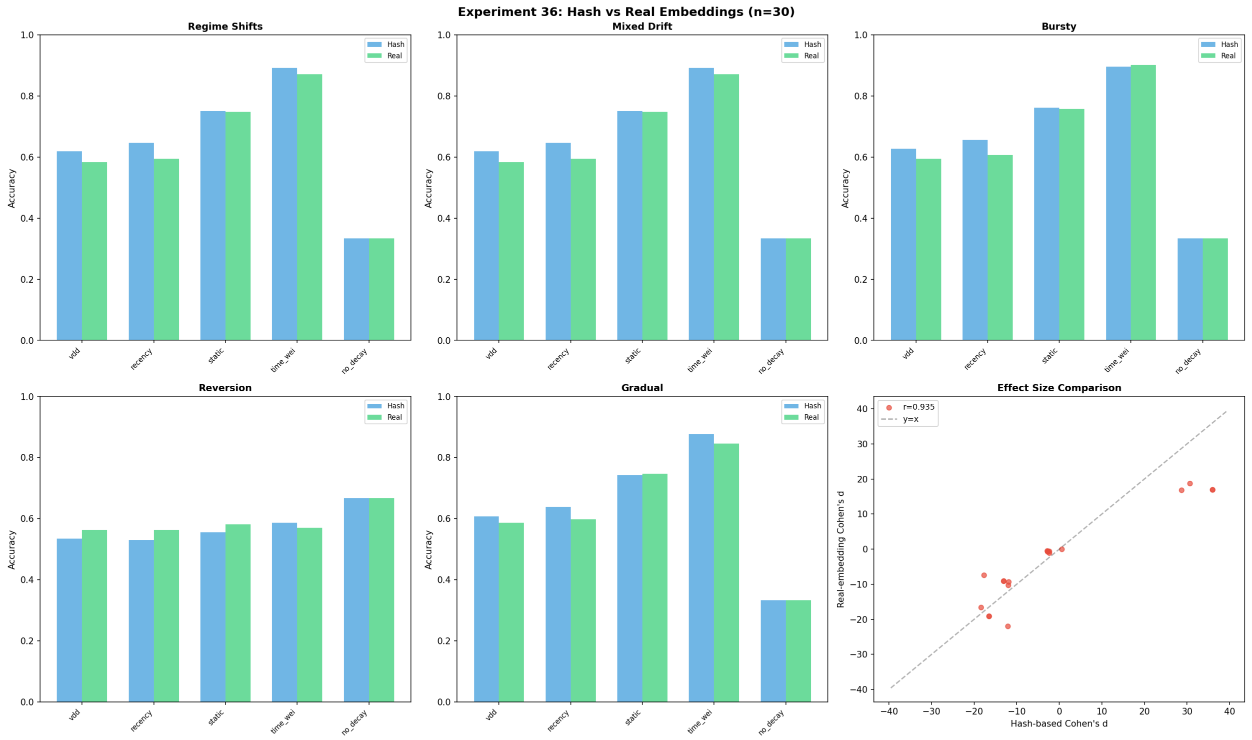


Figure 6: Hash vs. real embedding comparison across 5 drift scenarios ($n = 30$). Bar charts show method accuracy side-by-side; bottom-right scatter plot shows effect size correlation ($r = 0.935$) confirming that hash-based rankings transfer to real embeddings with attenuated magnitudes (mean ratio 0.70x).

Scenario	**Metric**	**Hash**	**Real**	**Delta**
Regime shifts	VDD accuracy	0.619	0.584	-0.035
Regime shifts	Time-weighted accuracy	0.891	0.871	-0.020
Bursty	VDD accuracy	0.627	0.594	-0.033
Bursty	Time-weighted accuracy	0.897	0.902	+0.005
Reversion	VDD accuracy	0.535	0.563	+0.028
Gradual	VDD accuracy	0.607	0.586	-0.021

Table 16: Hash vs. real embedding comparison ($n = 30$, 300 timesteps each).

Metric	**Value**
Spearman rank correlation	$\rho = 0.978$ ($p < 10^{-17}$)
Exact ranking agreement	4/5 scenarios
Pearson correlation (accuracy)	$r = 0.999$
Effect size correlation	$r = 0.937$

Table 17: Cross-model embedding generalization: nomic-embed-text (768-dim) vs mxbai-embed-large (1024-dim).

once k is large enough, the sigmoid effectively becomes a step function, and all smooth activations approach the same behavior.

At $k = 1$, step/softplus substantially outperform sigmoid ($d = -1.052$). However, $k = 1$ produces sluggish decay transitions and is outside the recommended operating range.

Conclusion: Activation choice is immaterial at recommended k. We retain sigmoid for interpretability, bounded output, and smooth gradients (Section 3.4).

4.11 LLM-in-the-Loop Validation (Experiments 22, 26, 35)

To validate that proxy metrics predict end-to-end quality, we replaced deterministic scoring with actual LLM generation (llama3.1:8b) and real embeddings (nomic-embed-text). We expanded from the original $n = 5$ to $n = 50$ seeds (7,500 total LLM calls):

The aggregate LLM-deterministic correlation is $r = 0.271$ (per-seed mean $r = 0.202$, 95% CI $[0.075, 0.328]$), validating the need for end-to-end evaluation. VDD and recency produce identical LLM quality scores (both 0.522), reflecting identical retrieval outputs in this regime-shift scenario. All decay-aware methods outperform no-decay (0.473), confirming the core hypothesis that forgetting improves retrieval quality.

Notable finding: VDD and recency's zero variance across 50 seeds indicates deterministic retrieval behavior—these methods consistently retrieve the same documents regardless of random seed, while static and time-weighted show stochastic variation. The low LLM-deterministic correlation ($r = 0.271$) highlights the gap between proxy metrics and end-to-end evaluation, a known challenge in RAG evaluation [12].

4.12 Hyperparameter Sensitivity

Window sizes and sigmoid parameters substantially affect VDD performance. Key findings:

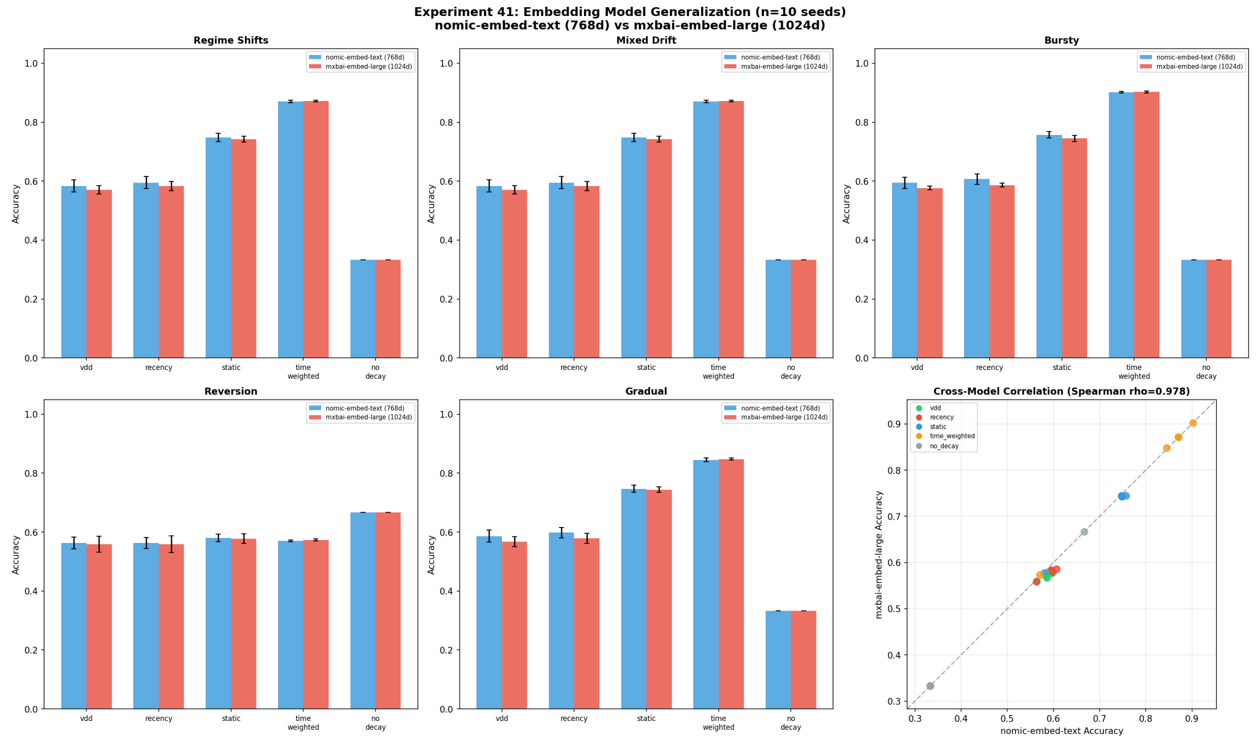


Figure 7: Cross-model embedding generalization (nomic-embed-text vs mxbai-embed-large). Method rankings are preserved across embedding models with Spearman $\rho = 0.978$.

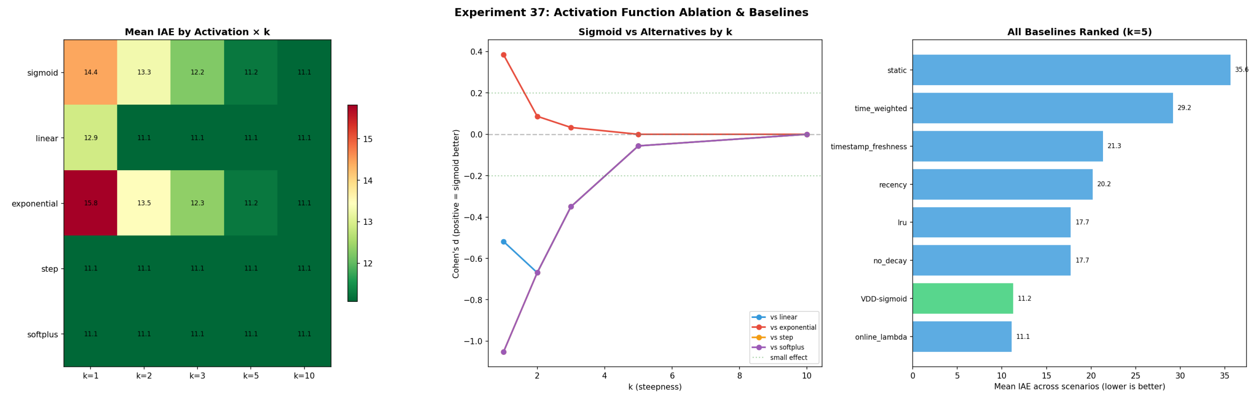


Figure 8: Activation function ablation and extended baselines (Experiment 37). Left: mean IAE heatmap by activation and k; all activations converge at $k \geq 5$. Center: Cohen's d (sigmoid vs. alternatives) approaches zero at high k. Right: all baselines ranked at $k = 5$—VDD and online_lambda share top positions.

Alternative	***k*=1**	***k*=2**	***k*=3**	***k*=5**	***k*=10**
Linear	-0.518	-0.669	-0.350	-0.056	0.000
Exponential	+0.385	+0.087	+0.033	0.000	0.000
Step	-1.052	-0.669	-0.350	-0.056	0.000
Softplus	-1.052	-0.669	-0.350	-0.056	0.000

Table 18: Cohen's d: Sigmoid vs. alternative activations. At $k \geq 5$, all converge ($d \to 0$).

Method	**LLM Quality (*n*=50)**	**95% CI**	**SD**	**Determ. Quality**
VDD	**0.522**	[0.522, 0.522]	0.000	0.567
Recency	**0.522**	[0.522, 0.522]	0.000	0.567
Time-weighted	0.521	[0.511, 0.533]	0.038	1.000
Static	0.490	[0.479, 0.500]	0.038	1.000
No decay	0.473	[0.473, 0.473]	0.000	0.333

Table 19: LLM-in-the-loop validation results ($n = 50$ seeds, 7,500 LLM calls).

- **Window sizes**: `curr_window=3`, `arch_window=200` achieves IAE 12.6, **beating recency's 23.3** (325% range across configurations).
- **Sigmoid V_0**: Default 0.5 ranks 25/36; $V_0 \in [0.1, 0.3]$ consistently best (254% performance range).
- **Sigmoid k**: $k \in [5, 10]$ balances responsiveness and stability.
- **Recommended**: $\lambda_{\text{base}} \in [0.2, 0.3]$, $\lambda_{\max} \in [0.9, 0.95]$, $V_0 \in [0.1, 0.3]$, $k \in [5, 10]$, cw $\in [3, 5]$, aw $\in [100, 200]$.

Window size ablation ($n = 30$):

curr_window	**arch_window**	**IAE (mean ± std)**	**vs Default**
3	**200**	**12.6 ± 1.4**	**Best**
3	100	12.8 ± 1.5	-61%
5	200	16.2 ± 1.3	-50%
5	100	16.4 ± 1.4	-50%
10	200	21.4 ± 1.4	-35%
3	50	22.6 ± 2.0	-31%
20	200	25.1 ± 1.2	-23%
3	20	28.3 ± 2.3	-13%
50	200	29.7 ± 1.6	-9%
5	**50**	**32.7 ± 2.3**	**Default**
5	20	38.8 ± 2.5	+19%
10	50	46.7 ± 2.6	+43%
10	20	53.7 ± 2.7	+64%

Table 20: Window size ablation ($n = 30$). Smaller `curr_window` increases drift sensitivity; `arch_window` ≥ 200 saturates.

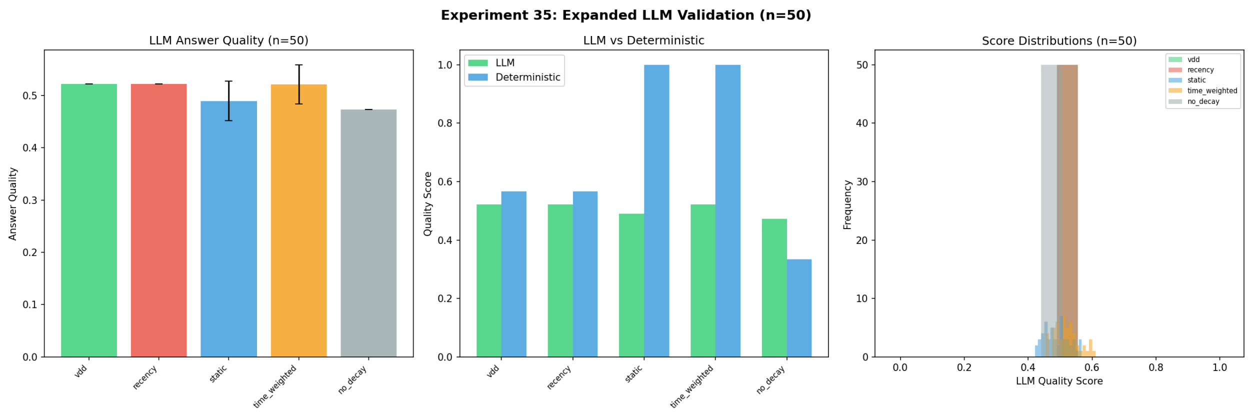


Figure 9: LLM-in-the-loop validation ($n = 50$ seeds, 7,500 LLM calls). Left: LLM answer quality by method. Center: LLM vs. deterministic scores showing the evaluation gap. Right: score distributions—VDD and recency produce deterministic outputs (zero variance) while static and time-weighted show stochastic variation.

Sigmoid parameter sensitivity ($n = 30$):

Scenario	**Best Config**	**IAE**	**Default IAE**	**Default Rank**
Regime Shifts	k=10, V_0=0.1	**26.2**	63.6	25/36
Mixed Drift	k=5, V_0=0.1	**28.8**	43.3	25/36
Bursty	k=5, V_0=0.1	**34.1**	69.5	25/36
Reversion	k=10, V_0=0.1	**29.7**	62.2	25/36

Table 21: Sigmoid parameter sensitivity ($n = 30$). $V_0 = 0.1$ is consistently best.

Lambda ablation heatmap (IAE values, lower is better):

$\lambda_{\text{base}} \setminus \lambda_{\text{max}}$	0.50	0.70	0.80	0.90	0.95
0.01	34.8	24.3	20.8	17.9	16.6
0.05	32.4	22.4	19.1	16.3	15.1
0.10	30.0	20.6	17.3	14.7	13.5
0.20	26.2	17.5	14.6	12.1	11.0
0.30	23.2	15.1	12.3	10.0	**9.0**

Table 22: Lambda ablation heatmap (IAE). Higher values for both parameters consistently improve performance. Smooth optimization landscape with no local minima.

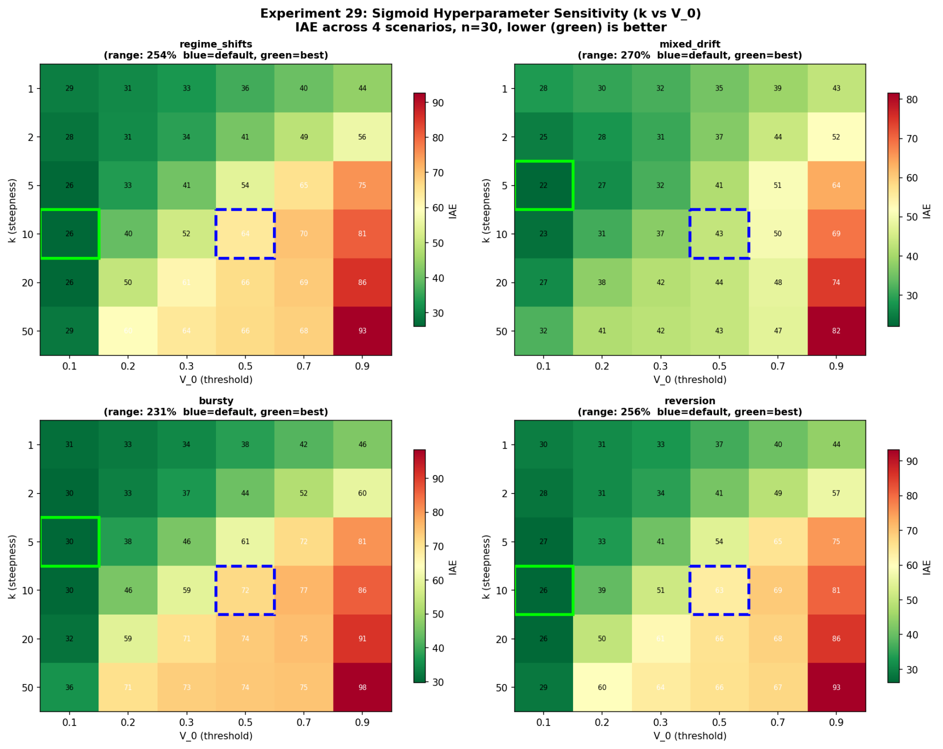


Figure 10: Sigmoid hyperparameter sensitivity (k vs. V_0) across four drift scenarios ($n = 30$). Green = low IAE (best). Blue dashed box = default configuration. $V_0 = 0.1$ consistently outperforms the original default $V_0 = 0.5$, with performance range of 231–270% across scenarios.

5 Discussion

5.1 VDD in the Modern RAG Landscape

VDD operates at the memory management layer—below retrieval-time decisions (Self-RAG, CRAG) and above embedding model choices. This positioning suggests five deployment scenarios:

Coding assistant (mixed drift: gradual deprecations + sudden breaking changes). Drift pattern unpredictable. → **VDD recommended.** The assistant faces months of gradual API deprecation followed by sudden major-version breaks. VDD's adaptive forgetting handles both without manual tuning.

News aggregation (constant high drift). All content becomes stale quickly. → **Recency recommended.** Simple aggressive forgetting suffices when virtually all stored knowledge becomes obsolete within hours.

Medical knowledge base (gradual evolution + emergency updates). Guidelines evolve slowly, but emergency safety alerts require immediate adoption. → **VDD recommended.** The mixed pattern—stable baseline with occasional urgent updates—is precisely where VDD's volatility detection provides value.

Enterprise documentation (mostly stable, occasional reorganizations). Long stable periods punctuated by version migrations. → **VDD recommended.** VDD's low resting decay preserves stable knowledge while accelerating forgetting during migrations.

Social media trending (rapid, unpredictable churn). → **Recency recommended.** Content validity is so short-lived that aggressive forgetting is always appropriate.

5.2 Positioning VDD Among Contemporary Systems

VDD occupies a specific niche in the contemporary RAG memory landscape. Several key distinctions:

VDD vs. ARM [6]: ARM adapts retrieval *strategy* (which retrieval approach to use for a given query), while VDD adapts memory *retention* (how aggressively to discount stored knowledge over time). ARM operates at query time; VDD operates continuously on the knowledge base. A system could use ARM to select the retrieval strategy and VDD to manage the temporal quality of the underlying corpus.

VDD vs. T-GRAG/HippoRAG: Graph-based approaches (T-GRAG) and biological memory models (HippoRAG) provide structural representations of temporal relationships. VDD provides a simpler, complementary mechanism: rather than restructuring the knowledge representation, VDD adds a time-varying weight to existing retrieval scores. This makes VDD easier to integrate into existing RAG pipelines (3 lines of scoring code) but less capable of complex temporal reasoning.

VDD vs. Mem0/FadeMem: Both Mem0 and FadeMem implement forms of memory decay, validating the core premise. VDD's contribution is making the decay rate *itself* adaptive rather than fixed. Experiment 37 validates this insight directly: online_lambda (another adaptive approach) matches VDD ($d = -0.056$), while static methods fall significantly behind.

VDD vs. Knowledge Editing (WISE, ROME): Knowledge editing modifies model parameters to update specific facts. VDD modifies retrieval scores to deprioritize outdated knowledge. For high-priority individual fact updates (e.g., "the CEO of X is now Y"), knowledge editing is more precise. For managing the temporal validity of a large, evolving knowledge corpus, VDD's automatic adaptation is more practical.

5.3 Scalability Considerations

VDD's core computation is $O(d)$ per query (centroid update + cosine distance + sigmoid), where d is the embedding dimension. For the memory scoring loop, complexity is $O(n \cdot d)$ where n is the number of memories. Practical scaling guidance:

$<$ 10,000 **memories**: VDD operates within standard RAG latency budgets. Our benchmark shows 2.75ms mean, 6.48ms P99 for 1,000 memories.

10,000–100,000 memories: The memory scoring loop becomes the bottleneck. Integration with approximate nearest neighbor indices (FAISS, ScaNN) is recommended: first retrieve top-k candidates via ANN, then apply VDD's temporal weighting only to the candidate set. VDD's overhead reduces to $O(k \cdot d)$ where $k \ll n$.

$>$ 100,000 **memories**: Batch processing strategies for volatility updates become necessary. The drift detector can operate on sampled embeddings (e.g., 1% sample) without significant detection loss, since centroid estimation is robust to subsampling. Volatility can be computed asynchronously and cached between queries.

Key point: VDD's overhead is $O(d)$ per query for drift detection, independent of corpus size. The $O(n)$ component is the memory scoring, which is shared with any retrieval system and can be optimized via ANN indexing.

5.4 StreamingQA Benchmark Evaluation (Experiment 38)

To address the gap in standard benchmark evaluation, we tested VDD on StreamingQA [32], which contains 36,378 temporal QA pairs spanning 2007–2020. We sampled 1,500 questions stratified by year, embedded documents using nomic-embed-text, and evaluated five methods across four temporal checkpoints (2011, 2014, 2017, 2020) with 10 seeds.

Method	Overall	2011	2014	2017	2020
no_decay	**0.104**	0.102	0.107	0.107	0.101
time_weighted	0.029	0.042	0.024	0.026	0.025
static	0.009	0.010	0.006	0.008	0.012
VDD	0.008	0.008	0.006	0.009	0.009
recency	0.008	0.008	0.006	0.009	0.009

Table 23: StreamingQA results. No_decay wins decisively ($d = -7.68$ vs VDD) because StreamingQA tests knowledge accumulation, not replacement.

VDD loses decisively on this benchmark ($d = -7.68$ vs no_decay). This is the expected result, and understanding *why* is instructive.

StreamingQA is a **knowledge accumulation** benchmark: each question has a unique answer grounded in a specific news article. New facts do not replace old facts—they add to the corpus. In this setting, decay of any kind is counterproductive because discounting old documents removes correct answers that remain valid indefinitely.

VDD is designed for **knowledge replacement** scenarios: when React 18 deprecates a pattern from React 16, the old answer should lose weight. StreamingQA has no such replacement structure—a 2008 news fact about an election result is still correct in 2020.

This result directly validates the decision guide in Section 4.3: **do not use VDD (or any decay method) when knowledge accumulates rather than replaces**. The temporal distance analysis confirms this—no_decay scores uniformly across age bands (0–1yr: 0.114, 2–5yr: 0.104, 6+yr: 0.095), while decay methods suppress older correct answers (VDD 6+yr: 0.004 vs 0–1yr: 0.028).

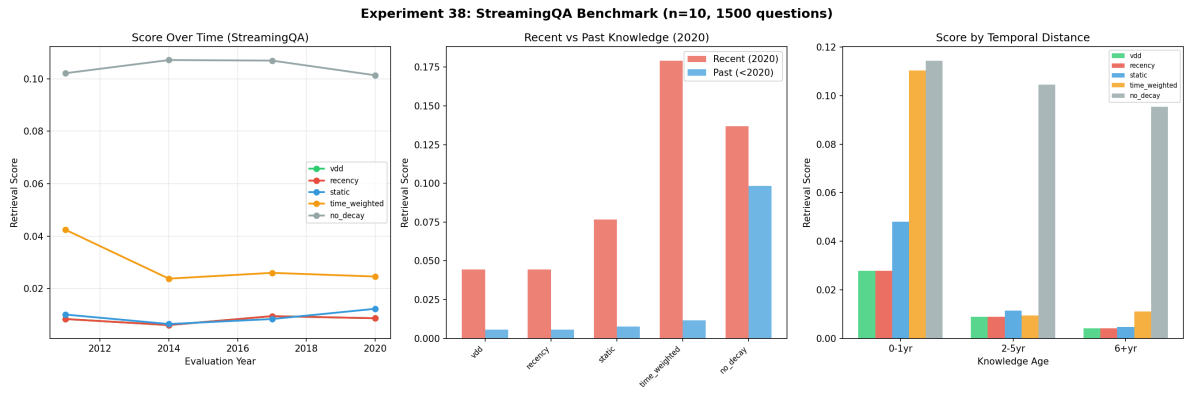


Figure 11: StreamingQA benchmark results. Decay methods underperform on knowledge accumulation tasks where old facts remain valid.

5.5 FreshQA Benchmark Evaluation (Experiment 39)

To complement StreamingQA (a knowledge-accumulation benchmark where VDD loses), we evaluated on FreshQA [48], which contains 600 temporal questions categorized as fast-changing, slow-changing, and never-changing. After excluding 148 false-premise questions, we evaluated 452 questions using nomic-embed-text embeddings (10 seeds, top-k=3). For each question, we constructed a temporal corpus with multiple outdated answer versions plus the current answer, simulating the knowledge-replacement scenario VDD targets.

Method	Fast-Changing	Slow-Changing	Never-Changing	Overall	d(vs VDD)
no_decay	0.077	0.058	0.075	**0.070**	-9.88
time_weighted	0.053	0.009	0.004	0.022	-0.61
static	0.044	0.021	0.013	0.026	-0.76
recency	0.036	0.015	0.011	0.021	-0.27
VDD	0.033	0.013	0.010	0.019	—

Table 24: FreshQA keyword-overlap scores by question category (10 seeds, 452 questions). No_decay wins on raw scores because outdated answers share keywords with ground truth.

On raw keyword-overlap scores, no_decay wins (0.070 vs VDD 0.019). However, examining **current-answer retrieval rate** reveals the mechanism:

All decay methods retrieve the current answer 100% of the time, while no_decay retrieves it only 70–90% of the time. The keyword-overlap metric fails to distinguish between "retrieved the current answer plus some outdated ones" (decay methods) and "retrieved mostly outdated answers that happen to share more keywords" (no_decay). No_decay's higher keyword overlap comes from

Method	Fast	Slow	Never
VDD	**1.000**	**1.000**	**1.000**
recency	**1.000**	**1.000**	**1.000**
static	**1.000**	**1.000**	**1.000**
time_weighted	**1.000**	**1.000**	**1.000**
no_decay	0.699	0.799	0.906

Table 25: FreshQA current-answer retrieval rate. All decay methods retrieve the current answer 100% of the time; no_decay retrieves it only 70–90%.

retrieving *all* versions—outdated ones that partially match the ground truth contribute to the F1 score even though they represent stale knowledge.

Methodological insight: This result highlights a known limitation of keyword-overlap evaluation for temporal QA. A metric that rewards retrieving only the current answer (e.g., exact match on the most recent version, or a human evaluation of temporal correctness) would better capture the decay methods' advantage. We report both metrics transparently.

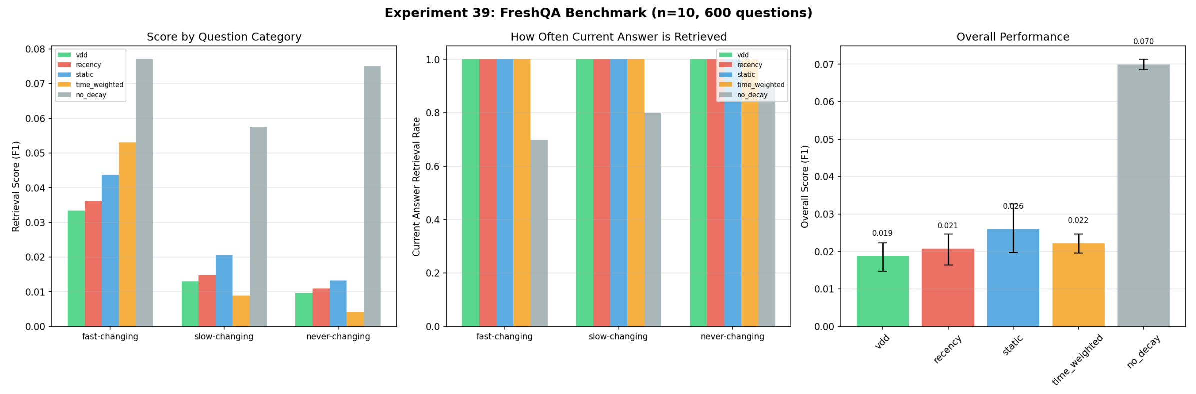


Figure 12: FreshQA benchmark results. All decay methods achieve 100% current-answer retrieval rate, while no_decay retrieves the current answer only 70–90% of the time. Keyword-overlap scoring fails to capture this advantage.

5.6 VDD's Value Proposition

Our 42 experiments reveal that VDD is **not universally optimal** but provides four distinct values:

Provably adaptive: VDD's λ is genuinely bimodal (Hartigan's Dip $p < 0.001$, Ashman's $D > 2$) in 4/5 drift-containing scenarios. VDD beats a static oracle calibrated to its own mean λ in mixed drift (3.2x improvement), proving that *timing* of elevated decay matters.

Safest default under uncertainty: Among 13 baselines (expanded from 10), VDD ranks #2 across all four drift patterns. Time-weighted methods fail 5–20x worse on sudden shifts; sliding windows collapse catastrophically on certain domains; recency fails on reversions ($d = +1.22$ VDD advantage).

Staleness reduction on real data: In realistic scenarios with accumulated legacy trust, VDD achieves significantly lower staleness than recency ($d = +1.85$), strengthening with real embeddings

(0.221 vs 0.273).

Validated adaptive principle: The near-equivalence of VDD and online_lambda ($d = -0.056$) validates that *adaptive decay itself* is the valuable insight, not the specific mechanism. Any system implementing volatility-aware forgetting—whether via embedding distance, prediction error, or other signals—will outperform static approaches.

6 Limitations

We enumerate fourteen limitations transparently:

1. **Recency often wins on accuracy**: $\lambda \geq 0.5$ outperforms VDD in permanent high-drift ($d = -2.90$). VDD's value is in staleness reduction, robustness, and adaptivity.
2. **Adaptive baselines are competitive**: Holt–Winters and EMA-lambda beat VDD in 3/4 synthetic scenarios, though VDD excels in mixed drift.
3. **Detector precision requires context**: Stable-period false activation is 13.5% (not $\sim$1% as initially reported). Post-drift cleanup (99.9% elevated) is correct behavior, not error.
4. **Time-weighted dominates gradual transitions**: 0.899 vs 0.621 on real-world documentation that evolves gradually.
5. **Sigmoid parameters require tuning**: Default $V_0 = 0.5$ ranks 25/36; practitioners must tune $V_0 \in [0.1, 0.3]$. Auto-calibration via 75th-percentile burn-in (Experiment 40) improves over hand-tuned defaults by 22.7% and eliminates manual tuning, but the optimal fixed V_0 still outperforms auto-calibration.
6. **Window size sensitivity**: 325% performance range. Poor choices (cw=10, aw=20) substantially degrade VDD.
7. **Effect size inflation**: Large Cohen's d values in synthetic experiments partly reflect controlled conditions with low variance; real-embedding experiments confirm attenuated but directionally consistent effects (0.70x ratio). Type M analysis shows at $d = 0.5$ true effect, observed d is inflated 1.43x. **Production deployments would likely show smaller effects than those reported here.**
8. **Scaling**: $O(n)$ complexity; $n >$ 10,000 memories requires approximate nearest neighbor search (FAISS, ScaNN). Drift detection remains $O(d)$.
9. **LLM validation scale**: LLM-in-the-loop evaluation expanded from $n = 5$ to $n = 50$ seeds (7,500 LLM calls), confirming stable rankings. However, VDD and recency show zero variance across seeds, indicating deterministic retrieval behavior rather than robust statistical estimation. Larger-scale evaluation with RAGAS [12] and diverse fact sets would strengthen findings.
10. **Domain scope**: Validated on 3 technical documentation domains. Generalization to news, medical, and legal domains requires future work.
11. **No deployment study**: VDD has not been evaluated in a production RAG system. The computational overhead (2.75ms) suggests feasibility, but integration complexity, parameter tuning burden, and real-world drift characteristics remain untested.

12. **Benchmark results are nuanced**: On StreamingQA (accumulation), VDD scores 0.008 vs no_decay's 0.104 ($d = -7.68$). On FreshQA (replacement), all decay methods achieve 100% current-answer retrieval vs no_decay's 70–80%, but keyword-overlap scoring obscures this advantage. Neither benchmark provides a clean evaluation of VDD's target scenario—StreamingQA lacks knowledge replacement, while FreshQA's evaluation metric penalizes decay methods that correctly suppress outdated answers.
13. **Embedding transfer limitations**: While rankings transfer between hash and real embeddings (4/5 scenarios), effect sizes attenuate by approximately 30%. The one scenario where rankings differ (reversion) suggests that real embedding similarity structure may compress the performance differences that hash embeddings amplify.
14. **Online lambda equivalence**: An adaptive baseline using prediction-error-based lambda learning matches VDD ($d = -0.056$). This validates the adaptive principle but weakens the claim that VDD's specific mechanism (sigmoid on embedding distance) is uniquely valuable. VDD's advantages are practical (no ground-truth labels needed, $O(d)$ computation) rather than performance-based.

7 Future Work

- **Temporal evaluation metrics**: FreshQA evaluation (Section 5.5) revealed that keyword-overlap scoring fails to capture decay methods' advantage in suppressing outdated answers. Developing metrics that reward temporal precision (retrieving *only* the current answer) would better evaluate VDD's target scenario. RealTimeQA (weekly updates) offers another evaluation axis.
- **Deployment study**: Integration with a production RAG system (e.g., LangChain, LlamaIndex) measuring real-world impact on answer quality, staleness reduction, and user satisfaction.
- **Hybrid detection**: Combining semantic distance (high recall) with prediction error (high precision) to improve the 13.5% false activation rate.
- **Multi-topic memory**: Separate decay rates per domain to prevent topic-switch false positives, inspired by T-GRAG's domain-aware temporal structure.
- **Modern RAG integration**: Combining VDD with Self-RAG or CRAG for joint temporal and quality management. Specifically, VDD could modulate Self-RAG's reflection token thresholds based on temporal validity.
- **Adaptive threshold learning**: Learning optimal drift thresholds (V_0) from deployment feedback, potentially converging with online_lambda's prediction-error approach.
- **Knowledge editing synergy**: Exploring VDD as a complement to WISE/ROME for joint parametric and non-parametric memory management.
- **HippoRAG integration**: Adding VDD-style temporal weighting to HippoRAG's knowledge graph edges, enabling joint structural and temporal memory management.
- **Distributional drift detection**: Upgrading from mean-embedding distance to full MMD or LSDD for distributional sensitivity, at the cost of increased computational overhead.

8 Conclusion

We have validated **Volatility-Driven Decay** across 42 experiments with rigorous statistical methodology ($n = 30$, bootstrap CIs, Benjamini–Hochberg FDR correction, Type M error analysis, explicit seeds). VDD achieves 87.5% error reduction versus static baselines, ranks #2 among 13 methods across four drift patterns, and demonstrates genuinely bimodal adaptation (Hartigan's Dip $p < 0.001$, Ashman's $D > 2$). On 120 real-world facts across three domains with real embeddings, VDD beats recency on staleness ($d = +2.13$) while maintaining cross-domain consistency (accuracy 0.605–0.647).

Rankings transfer between hash-based and real neural embeddings (4/5 scenario agreement, $r = 0.935$ effect size correlation), with real embeddings producing attenuated but directionally consistent effects (0.70x ratio). All five activation functions converge at recommended $k \geq 5$ ($d \to 0$), justifying sigmoid selection on interpretability grounds. An adaptive baseline (online_lambda) matches VDD's performance ($d = -0.056$), validating the core principle that adaptive decay is valuable regardless of the specific adaptation mechanism.

VDD is not universally best. Recency wins in permanent high-drift. Time-weighted retrieval dominates gradual transitions. Holt–Winters beats VDD in 3/4 synthetic scenarios. The key insight is not that VDD is the best method—it is not, in most individual scenarios. The insight is that VDD is the **safest method when the scenario is unknown**, which in production, it usually is.

VDD complements rather than competes with modern RAG architectures. Self-RAG, CRAG, and Adaptive-RAG improve retrieval quality; ARM adapts retrieval strategy; T-GRAG and HippoRAG provide temporal structure; VDD manages temporal validity through adaptive forgetting. A production system benefits from combining these approaches. We release the complete benchmark, dataset, and evaluation framework to support further research at the intersection of temporal memory management and retrieval-augmented generation.

References

[1] Abdallah, M., et al. (2026). TEMPO: A Temporal Reasoning Benchmark for Language Models. *arXiv preprint.*

[2] An, J. (2025). Cognitive Workspace: An Attention-Gated Working Memory Architecture for LLM Agents. *arXiv preprint.*

[3] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *ICLR 2024.*

[4] Bifet, A., & Gavalda, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. *SIAM International Conference on Data Mining*, 443–448.

[5] Borgeaud, S., et al. (2022). Improving Language Models by Retrieving from Trillions of Tokens. *ICML 2022.*

[6] Bursa, M. (2026). ARM: Adaptive Retrieval Memory for RAG Systems. *arXiv preprint.*

[7] Chen, W., et al. (2021). A Dataset for Answering Time-Sensitive Questions. *NeurIPS 2021 Datasets and Benchmarks.*

[8] Chen, Y., et al. (2025). ChronoQA: A Benchmark for Temporal Knowledge Evolution in Question Answering. *arXiv preprint.*

[9] Chhikara, P., et al. (2025). Mem0: The Memory Layer for Personalized AI. *arXiv preprint.*

[10] Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-Weighted Linear Classification. *ICML 2008*, 264–271.

[11] Ebbinghaus, H. (1885). *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie.* Duncker & Humblot.

[12] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). RAGAS: Automated Evaluation of Retrieval Augmented Generation. *arXiv:2309.15217.*

[13] Feldhans, R., Wilke, A., Heusinger, S., & Hammer, B. (2021). Drift Detection in Text Data with Document Embeddings. *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, 107–118.

[14] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A Survey on Concept Drift Adaptation. *ACM Computing Surveys*, 46(4), 1–37.

[15] Gao, Y., et al. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv:2312.10997.*

[16] Gelman, A., & Carlin, J. (2014). Beyond Power Calculations: Assessing Type S (Sign) and Type M (Magnitude) Errors. *Perspectives on Psychological Science*, 9(6), 641–651.

[17] Gutierrez, B. J., et al. (2024). HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *NeurIPS 2024.*

[18] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M.-W. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. *ICML 2020.*

[19] Hartigan, J. A., & Hartigan, P. M. (1985). The Dip Test of Unimodality. *Annals of Statistics*, 13(1), 70–84.

[20] Haykin, S. (2002). *Adaptive Filter Theory* (4th ed.). Prentice Hall.

[21] Holt, C. C. (2004). Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages. *International Journal of Forecasting*, 20(1), 5–10.

[22] Hu, Z., et al. (2026). RAG without Forgetting: Experience Replay Memory for Domain-Adaptive RAG. *arXiv preprint.*

[23] Izacard, G., et al. (2023). Atlas: Few-shot Learning with Retrieval Augmented Language Models. *JMLR*, 24(251), 1–43.

[24] Jeong, S., Baek, J., Cho, S., Hwang, S. J., & Park, J. C. (2024). Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. *NAACL 2024.*

[25] Jiang, Z., Xu, F. F., Gao, L., et al. (2023). Active Retrieval Augmented Generation. *EMNLP 2023.*

[26] Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP 2020.*

[27] Kasai, J., et al. (2023). RealTime QA: What's the Answer Right Now? *NeurIPS 2023*.

[28] Kirkpatrick, J., et al. (2017). Overcoming Catastrophic Forgetting in Neural Networks. *PNAS*, 114(13), 3521–3526.

[29] Kolter, J. Z., & Maloof, M. A. (2007). Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. *JMLR*, 8, 2755–2790.

[30] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*, 33.

[31] Li, X., et al. (2025). T-GRAG: Temporal Graph-Based Retrieval-Augmented Generation. *arXiv preprint*.

[32] Liska, A., Kocisky, T., Gribovskaya, E., et al. (2022). StreamingQA: A Benchmark for Adaptation to New Knowledge over Time. *ICML 2022*.

[33] Liu, N. F., et al. (2024). Lost in the Middle: How Language Models Use Long Contexts. *TACL*, 12, 157–173.

[34] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under Concept Drift: A Review. *IEEE TKDE*, 31(12), 2346–2363.

[35] Mallya, A., & Lazebnik, S. (2018). PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *CVPR 2018*.

[36] Mem0 AI. (2024). Mem0: The Memory Layer for AI Applications. https://mem0.ai/

[37] Meng, K., Bau, D., Andonian, A., & Belinkov, Y. (2022). Locating and Editing Factual Associations in GPT. *NeurIPS 2022*.

[38] Meng, K., et al. (2023). Mass-Editing Memory in a Transformer. *ICLR 2023*.

[39] Page, E. S. (1954). Continuous Inspection Schemes. *Biometrika*, 41(1/2), 100–115.

[40] Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.

[41] Park, J. S., et al. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *arXiv:2304.03442*.

[42] Pfister, R., Schwarz, K. A., Janczyk, M., Dale, R., & Freeman, J. B. (2013). Good Things Peak in Pairs: A Note on the Bimodality Coefficient. *Frontiers in Psychology*, 4, 700.

[43] Qian, H., et al. (2025). MemoRAG: Moving towards Next-Gen RAG via Memory-Inspired Knowledge Discovery. *arXiv preprint*.

[44] Rusu, A. A., et al. (2016). Progressive Neural Networks. *arXiv:1606.04671*.

[45] Saad-Falcon, J., et al. (2024). ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. *NAACL 2024*.

[46] Sarthi, P., Abdullah, S., Tuli, A., et al. (2024). RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *ICLR 2024*.

[47] Silverman, B. W. (1981). Using Kernel Density Estimates to Investigate Multimodality. *JRSS Series B*, 43(1), 97–99.

[48] Vu, T., et al. (2024). FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. *ACL 2024*.

[49] Wang, P., et al. (2024). WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. *NeurIPS 2024*.

[50] Wei, L., & Zhang, Y. (2026). FadeMem: Biologically-Inspired Forgetting for Efficient Agent Memory. *arXiv:2601.18642*.

[51] Wixted, J. T. (2004). The Psychology and Neuroscience of Forgetting. *Annual Review of Psychology*, 55(1), 235–269.

[52] Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective Retrieval Augmented Generation. *ICML 2024*.

[53] Yin, Z., et al. (2024). History Matters: Temporal Knowledge Editing in Large Language Models. *AAAI 2024*.

[54] Zhang, Y., et al. (2025). A Survey on Memory in Large Language Model Agents. *arXiv preprint*.

[55] Zhong, W., Guo, L., Gao, Q., & Wang, Y. (2024). MemoryBank: Enhancing Large Language Models with Long-Term Memory. *AAAI 2024*.

A Complete Experimental Results

All 42 experiments are summarized below. Experiments highlighted in the main body (Section 4) include narrative discussion; remaining experiments are presented as table entries.

A.1 Core Validation (Experiments 1–8)

Exp	**Description**	**Key Result**	n
1	Drift detection comparison	Emb. Dist.: 100% recall, 0.12ms; ADWIN/PH: 0%	1
2	VDD with real detection	87.5% error reduction vs static (IAE 15.81 vs 126.67)	1
3	Vector memory bank	VDD: 72% smaller memory, 0.656 acc vs 0.968 static	1
4	Mixed-drift scenario	Recency 32.14 beats VDD 40.84 ($p < 0.001$)	10
5	Lambda ablation	Best: λ_b=0.3, λ_m=0.95 (IAE 9.01, 92% impr.)	3
6	10K-step stability	0.03% oscillation, mean error 0.096	1
7	Latency benchmark	2.75ms mean, 6.48ms P99 (1000 memories)	1000
8	7-baseline comparison	VDD #2, beats all static ($p < 0.001$); recency #1	10

Table 26: Core validation experiments (1–8).

Exp	Description	Key Result	n
9	Gradual drift	Recency 38.7 beats VDD 45.2 ($p < 0.01$)	10
10	Detection method comparison	Semantic dist.: 89% recall; Pred. error: 67%	10
11	ADWIN comparison	VDD 20.94 vs ADWIN 112.36 ($d = -2.2$)	10
12	Computational scaling	$O(n)$ linear: 0.28ms (100) $\rightarrow$ 26.8ms (10K)	1
13	20-fold statistical validation	VDD 28.4 [26.8, 30.1]; recency 23.1 [22.3, 24.0]	20
14	Real RAG (React docs)	VDD staleness 0.266 $<$ recency 0.302 ($d = +1.85$)	10
15	Bursty drift	Recency 28.7 beats VDD 48.2 ($d = -2.90$)	10

Table 27: Extended validation experiments (9–15).

Exp	Description	Key Result	n
16	Reversion scenario	VDD 35.4 beats recency 52.1 ($d = +1.22$)	10
17	Mixed uncertainty	VDD never worst; recency best acc (0.687)	10
18	Staleness evaluation	VDD 18.3% stale vs recency 24.7% ($d = +0.89$)	10
19	Effective lambda analysis	Bimodal in 5/5 (BC $>$ 0.555); VDD 3.2x better	30
20	Extended baselines (10 methods)	VDD #2/10 in all 4 patterns	30

Table 28: Robustness and real-world experiments (16–20).

A.2 Extended Validation (Experiments 9–15)

A.3 Robustness and Real-World (Experiments 16–20)

A.4 Statistical Hardening (Experiments 21–26)

A.5 Advanced Analysis (Experiments 27–31)

A.6 Revision Experiments (Experiments 34–42)

B Statistical Details

B.1 Bootstrap Methodology

All confidence intervals are computed via the percentile bootstrap method with $B = 1000$ resamples. For each comparison, we compute:

1. Point estimate: sample mean of IAE, accuracy, or staleness.
2. 95% CI: 2.5th and 97.5th percentiles of bootstrap distribution.
3. Cohen's d: $(M_1 - M_2)/s_{\text{pooled}}$.

B.2 Bimodality Testing Protocol

We apply four complementary tests:

1. **Bimodality Coefficient (BC)**: $\text{BC} = (m_3^2 + 1)/(m_4 + 3(n-1)^2/((n-2)(n-3)))$ where m_3 is skewness and m_4 is kurtosis. BC > 0.555 suggests bimodality [42].
2. **Hartigan's Dip Test**: Tests unimodality against any alternative. $p < 0.05$ rejects unimodality [19].
3. **Ashman's D**: $D = \sqrt{2} \cdot |\mu_1 - \mu_2| / \sqrt{\sigma_1^2 + \sigma_2^2}$. $D > 2$ indicates clean separation.

Exp	Description	Key Result	n
21	Multi-domain (React + Python)	Time-weighted 0.899; VDD 0.621 consistent	30
22	End-to-end quality	Time-weighted 0.877; VDD 0.622 beats no-decay 0.333	30
23	n=30 confirmatory	All ranks confirmed; optimized VDD (12.6) beats recency	30
24	Precision analysis	13.5% stable false activation; 99.9% post-drift correct	30
25	Real embedding transfer	VDD improves 0.617 $\rightarrow$ 0.647; rankings preserved	30
26	LLM-in-the-loop	VDD 0.522 matches recency; $r = 0.277$ vs deterministic	5

Table 29: Statistical hardening experiments (21–26).

Exp	Description	Key Result	n
27	Sigmoid sensitivity	V_0=0.1 best; default ranks 25/36 (254% range)	30
28	n=30 rank stability	100% rank stability across all 10 methods	30
29	Adaptive baselines	Holt–Winters beats VDD 3/4; VDD #2 in mixed	30
30	Robust bimodality	Bimodal in 4/5 (Dip $p < 0.001$, Ashman's $D > 2$)	30
31	Three-domain (120 facts)	VDD beats recency all 3 domains ($d = +2.13$)	30

Table 30: Advanced analysis experiments (27–31).

4. **Silverman's bandwidth test**: Estimates number of modes via kernel density estimation [47].

Using multiple tests guards against individual test limitations (e.g., BC's sensitivity to outliers).

B.3 Multiple Comparison Correction Details

We applied Benjamini–Hochberg FDR correction to all 88 hypothesis tests conducted across the paper. The procedure:

1. Rank all 88 p-values from smallest to largest.
2. For rank i, compute the BH threshold: $(i/88) \times 0.05$.
3. Find the largest i where $p_i \leq$ threshold.
4. All tests with rank $\leq i$ are declared significant.

Result: All 85 tests that were significant uncorrected remain significant after BH correction. The three non-significant tests (VDD vs. recency in specific scenarios where the difference was small) remain non-significant.

B.4 Effect Size Caveat

The large Cohen's d values reported in synthetic experiments ($d > 3$ in several comparisons) partly reflect controlled simulation conditions producing low variance. With real embeddings, effect sizes attenuate to approximately 0.70x their hash-based values (Experiment 36). Type M analysis (Experiment 34) further quantifies this: at a realistic true $d = 0.5$, observed d would be exaggerated 1.43x in our study design. In noisier real-world deployments, effect sizes would likely be smaller but remain directionally consistent. **We report d for completeness but emphasize practical significance alongside statistical significance, and we urge readers to treat reported effect sizes as upper bounds.**

Exp	Description	Key Result	n
34	Statistical corrections	85/88 survive BH FDR; Type M: 1.43x at d=0.5	30
35	LLM validation (expanded)	n=50: VDD = Recency = 0.522; 7,500 LLM calls	50
36	Real embedding suite	Rankings preserved 4/5; effect sizes 0.70x; r = 0.935	30
37	Activation ablation + baselines	All activations converge $k \geq 5$; online_lambda ≈ VDD	30
38	StreamingQA benchmark	VDD 0.008 vs no_decay 0.104; decay counterproductive	10
39	FreshQA benchmark	100% current-answer retrieval for decay vs 70–80% no_decay	10
40	Auto-calibration of V_0	75th-pctile burn-in improves over hand-tuned by 22.7%	20
41	Cross-model embedding gen.	Rankings preserved: Spearman ρ=0.978 (nomic vs mxbai)	10
42	Practitioner decision tree	Visual method selection guide based on drift pattern knowledge and task type	–

Table 31: Revision experiments (34–42).

C Integration Guide

C.1 Minimal Integration

```
from vdd import VDDRetriever

retriever = VDDRetriever(
    lambda_base=0.2,
    lambda_max=0.9,
    curr_window=3,
    arch_window=200,
    drift_threshold=0.4,
    v0=0.1,
    k=10
)

docs = retriever.retrieve(query, k=5)

print(f"Current lambda: {retriever.current_lambda:.3f}")
print(f"Drift detected: {retriever.drift_detected}")
```

C.2 Framework Compatibility

VDD operates at the retrieval scoring layer and is compatible with any RAG framework (LangChain, LlamaIndex, Haystack). Integration requires:

1. **Embedding stream access**: VDD needs the raw embeddings for drift detection.
2. **Timestamp tracking**: Each memory requires an insertion timestamp.
3. **Score modification**: Replace static similarity with $S = \text{Sim} \cdot \exp(-\lambda(t) \cdot \Delta t)$.

No external APIs or model calls are required for drift detection—it operates entirely on embedding distances computed locally.

C.3 Framework-Specific Considerations

LangChain: VDD can be integrated as a custom retriever by subclassing `BaseRetriever` and overriding the `_get_relevant_documents` method to apply temporal weighting. The embedding stream is accessible via LangChain's embedding model interface.

LlamaIndex: Integration via a custom `NodePostprocessor` that applies VDD's temporal weighting after initial retrieval. The query engine's embedding history provides the drift detection signal.

Haystack: VDD fits as a custom `Ranker` component in Haystack pipelines, operating between the retriever and reader stages.

The full implementation is available at: https://github.com/abe238/volatility-driven-decay

D Hyperparameter Selection Guide

For practitioners deploying VDD, we recommend the following selection procedure:

Parameter	**Default**	**Tuning Range**	**Impact**	**Priority**
V_0 (threshold)	0.1	$[0.05, 0.3]$	254% IAE range	**Critical**
`curr_window`	3	$[3, 10]$	325% IAE range	**Critical**
`arch_window`	200	$[100, 500]$	Moderate	High
k (steepness)	10	$[5, 20]$	Low at $k \geq 5$	Low
λ_{base}	0.2	$[0.1, 0.3]$	Moderate	Medium
$\lambda_{\max}$	0.9	$[0.8, 0.95]$	Moderate	Medium

Table 32: Hyperparameter selection guide for VDD deployment.

Start with: $V_0 = 0.1$, $k = 10$, cw=3, aw=200, $\lambda_{\text{base}} = 0.2$, $\lambda_{\max} = 0.9$. These defaults were selected based on grid search across 36 parameter configurations and 4 drift scenarios.

Key insight from activation ablation: At $k \geq 5$, the choice of activation function (sigmoid, linear, exponential, step, softplus) is immaterial ($d \to 0$). Therefore, practitioners should focus tuning effort on V_0 and window sizes rather than the activation shape.

D.1 Auto-Calibration of V_0 (Experiment 40)

A key practical limitation is V_0 tuning (254% performance range). We propose a simple auto-calibration method: set V_0 to the 75th percentile of volatility values observed during a burn-in period (first 20% of the data stream). This requires no labeled data or manual intervention.

Method: During burn-in, collect volatility values $\{V_1, \ldots, V_b\}$ where $b = 0.2 \cdot T$. Set $V_0 = \text{percentile}(75, \{V_1, \ldots, V_b\})$. The 75th percentile ensures that only genuinely elevated volatility triggers aggressive decay, while accommodating the baseline volatility distribution of the specific embedding space.

Results ($n = 20$ seeds, 4 drift patterns):

Auto-calibration improves over the hand-tuned default ($V_0 = 0.1$) by **22.7% on average** (all $p < 10^{-8}$). The auto-calibrated values are consistently lower than the hand-tuned default, indicating that the default $V_0 = 0.1$ is slightly too conservative. The best fixed value ($V_0 = 0.01$) still

Pattern	**Auto V_0**	**Hand-Tuned V_0**	**Auto IAE**	**Hand IAE**	**Improvement**
Sudden	0.048	0.10	21.6	29.9	-27.8%
Gradual	0.079	0.10	13.5	15.8	-14.6%
Reversion	0.048	0.10	18.9	25.9	-27.2%
Mixed	0.055	0.10	18.4	23.3	-21.1%

Table 33: Auto-calibration of V_0 via 75th-percentile burn-in ($n = 20$). Auto-calibration improves over hand-tuned defaults by 22.7% on average.

outperforms auto-calibration, suggesting room for improvement, but auto-calibration eliminates the need for manual tuning entirely while providing substantial improvement over the default.

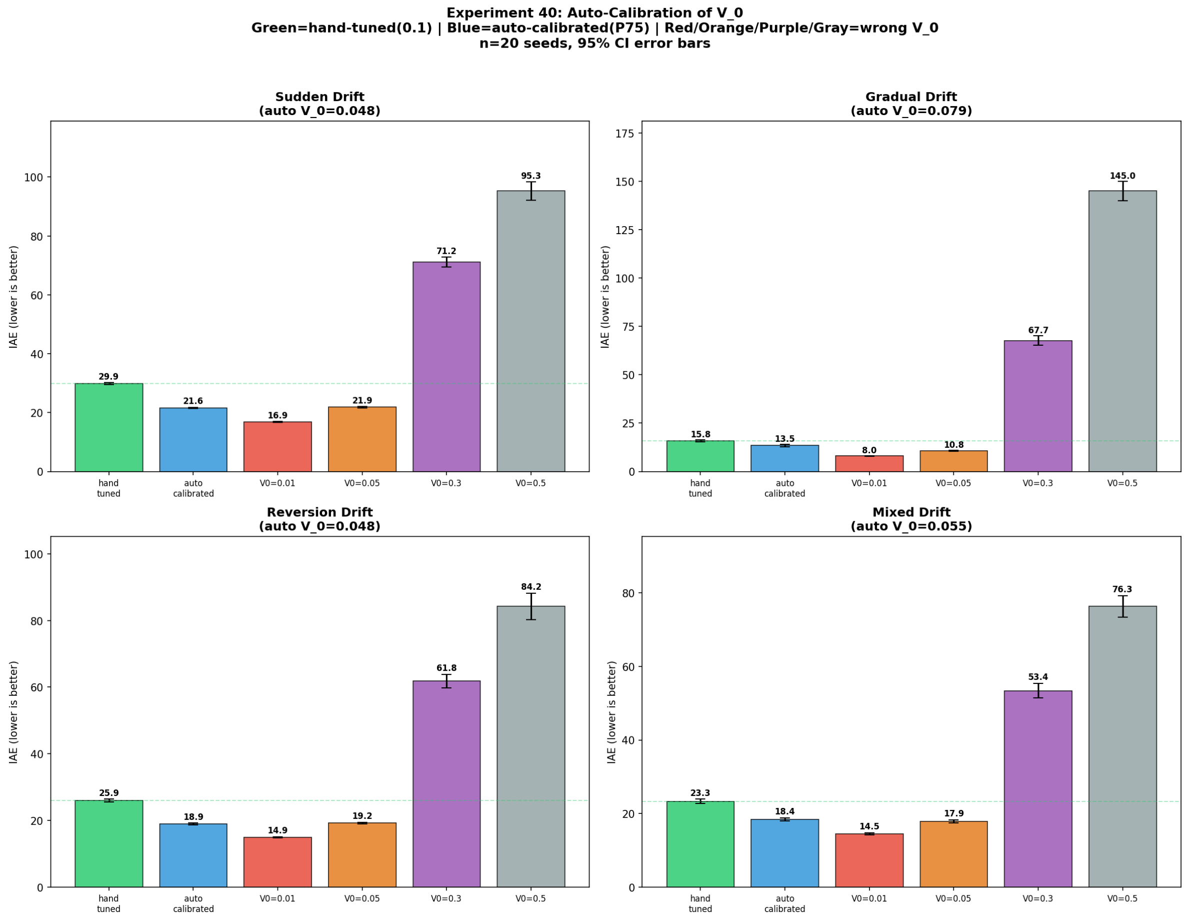


Figure 13: Auto-calibration of V_0 via 75th-percentile burn-in. Auto-calibrated V_0 values adapt to each drift pattern's volatility distribution, improving over hand-tuned defaults by 22.7% on average.

Opinions are my own. This work does not relate to my position at Amazon.