AgentDriftBench: Measuring Adaptation Dynamics of LLM Agents under Mid-Evaluation Environment Drift

Anonymous authors
Paper under double-blind review

Abstract

Deployed LLM agents operate in environments that change underneath them: API parameters are renamed, required fields appear, tools are deprecated—often mid-task. Static benchmarks cannot see the resulting failure mode: an agent may score well on a fixed suite yet collapse under a schema change and recover slowly, partially, or never. We introduce AgentDriftBench, a benchmark that injects controlled, mid-evaluation environment drift into multi-segment tool-use episodes and measures whether and how agents recover. AgentDriftBench contributes: (1) a drift engine with five drift families applied at segment boundaries, shipped with a treated-vs-control validity protocol (drift halves target-tool hit rate: 0.24 vs. 0.47, $z = -3.4$); (2) an adaptation-centric metric suite—post-drift success-curve AUC, Recovery Steps, Oscillation Index, and recovery rate—that scores recovery trajectories rather than end states; and (3) a controlled study of memory (scratchpad) and drift-notice prompting. Evaluating six Qwen models (0.5B–7B) on 100 six-segment episodes, we find: (i) a significant scale gradient in absolute adaptation (0.5B AUC 0.207 < 1.5B 0.312, p=0.0001), but normalized recovery is nearly flat—the absolute gap is largely baseline capability; (ii) memory value flips sign across models: scratchpad significantly hurts 0.5B (p=0.004) and helps Coder-3B (p=0.011), with Qwen3-0.6B borderline (p=0.058); (iii) drift notices backfire—conditionally: on 1.5B the notice reduces adaptation (p=0.009), and a budget–quality decomposition shows the harm tracks reduced tool-call budgets, not information content; (iv) drift without an error signal (D4) creates persistent deficits (up to −0.085, p<0.0001) that error-driven adaptation cannot cover. These phenomena are invisible to static benchmarks; AgentDriftBench makes adaptation dynamics a first-class measurement target.

1 Introduction

Static benchmarks answer one question: can the agent succeed? They cannot distinguish an agent that never understood the environment from one that was knocked down by a change and recovered. In deployment, the environment is not static: providers rename API parameters, add required fields, deprecate endpoints, and revise documentation while an agent is operating. An agent that scores 0.45 on a static suite may collapse under a mid-flight schema change and never recover—or recover in one step—and the static score is blind to the difference.

AgentDriftBench makes the environment itself the experimental variable. Each evaluation episode is a six-segment tool-use episode; at segment boundaries the benchmark injects controlled drift events from five families (parameter rename, added required parameter, tool deprecation/replacement, enum semantic shift, documentation update), and the agent must continue solving tasks against the changed environment. We score the recovery trajectory—not a single end-state success rate. Section 3 summarizes the design.

Three design commitments distinguish the benchmark:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1. Adaptation-centric metrics. Post-drift success-curve AUC, Recovery Steps, and Oscillation Index score the trajectory after a drift, with Pre-drift SR as the capability baseline and Normalized Recovery (AUC/pre-drift) decoupling "recovers well" from "was already strong."
2. Validity as a first-class methodology. Drift validity is certified by treated-vs-control comparisons: we ship the validators (V1–V4), the counterfactual protocol, and a list-position fairness check as part of the benchmark (§3).
3. Mechanism, not just scores. Every intervention effect (memory, notice) is decomposed into a budget channel (does the intervention change how many tool calls the agent makes?) and a quality channel (at equal call counts, does it choose better?). The decomposition closes exactly (< 0.001 residual) for both interventions across all models, revealing that "harm" and "benefit" travel through the same behavioral plumbing (§3.7, §5).

Contributions.

1. The first LLM-agent benchmark, to our knowledge, centered on post-drift recovery trajectories with certified drift validity: within-episode drift injection at segment boundaries plus recovery-trajectory measurement.
2. A metric suite (Adaptation AUC, Recovery Steps, Oscillation Index, recovery rate, + Pre-drift SR, Post-drift SR@k, Normalized Recovery) with formal definitions and a paired significance protocol (two-sided sign-flip permutation, 20k, episode-level).
3. Five drift families (D1 parameter rename $\times$2-weighted, D2 added required param, D3 tool deprecation, D4 enum semantic shift, D5 documentation update) with a validity-verified drift engine, plus memory and drift-notice control axes.
4. A systematic evaluation: 100 episodes $\times$ 6 segments, 6 models (0.5B$\rightarrow$7B, instruct/base/coder) $\times$ 4 configurations, with a no-drift counterfactual, a D4-dense paired experiment, and a unified significance protocol.
5. A budget–quality decomposition framework that uniformly explains the sign flips of memory and notice effects across models (linked to behavior via the budget channel vs. quality channel; exact closure; replicates across pools)—a reusable analysis tool for the hidden costs of prompt-level interventions.

2 Related Work

Agent benchmarks. AgentBench (Liu et al., 2024), τ-bench (Yao et al., 2024), τ^2-bench (Barrès et al., 2025), the Berkeley Function Calling Leaderboard (Patil et al., 2025), and GAIA (Mialon et al., 2023) evaluate agents under static environments: the toolset, tool schemas, and documentation do not change during evaluation. Our episodes are built from the hermes-function-calling corpus (Nous Research, 2024) and executed against a deterministic mock registry.

Closest work.

- Assidiqi et al. (2026, IEEE Access) (Assidiqi et al., 2026): a robustness benchmark perturbing schemas/policies/toolsets on τ^2-bench retail. As far as we can verify, they measure terminal degradation (robustness framing) rather than recovery trajectories, with no memory or notice axes.
- PACE-Bench (2026) (Zhan et al., 2026): recovery in physical/code domains, not tool environments.
- LifelongAgentBench (2025) (Zheng et al., 2025): lifelong learning across tasks; no within-episode drift injection.

Static perturbation benchmarks. RoTBench (Ye et al., 2024) (static perturbation levels), StableToolBench (Guo et al., 2024) (stability of the benchmark itself), MCP-Bench (Wang

Table 1: The five drift families. D1 is sampled at 2× weight.

Type	Mechanism	Example
D1 param rename	parameter renamed; old name silently fails	camera_id → device_id
D2 added required param	new required field appears	get_weather now requires units
D3 tool deprecation	tool removed; semantic equivalent under a new name/schema	search_flights → flight_search_v2
D4 enum semantic shift	doc appended: "Default/-expected value is now 'X"'	thermostat mode expected away instead of auto
D5 documentation update	tool description updated to reflect actual behavior	description now states the correct default

et al., 2025) (static multi-tool settings). These perturb before evaluation; none measures within-episode recovery.

Memory-related. MemEvoBench (Xie et al., 2026), KnowU-Bench (Chen et al., 2026): mobile-agent memory and personalization evaluation, not environment drift.

Non-stationarity in adjacent fields. Environment change during evaluation is the norm in non-stationary RL and concept-drift research, where adaptation speed and recovery are standard targets (Gama et al., 2014); WILDS (Koh et al., 2021) established distribution-shift benchmarks for supervised models. Continual-learning evaluations measure backward transfer (forgetting) under task sequences. AgentDriftBench imports this concern into LLM-agent evaluation, where the environment—not the task distribution—is what shifts, and the agent must re-ground its tool usage online.

Positioning. To our knowledge no prior benchmark (a) injects drift mid-evaluation with per-segment control, (b) scores recovery trajectories rather than end states, (c) crosses drift with memory and notice interventions, or (d) ships a treated-vs-control validity protocol for the drift itself.

3 AgentDriftBench Design

3.1 Task domain

Multi-segment tool-use episodes built from hermes-function-calling-v1 (Nous Research, 2024) (1,893 examples, 2,598 distinct tools). Each episode = 6 segments; each segment is a multi-turn tool-calling task with a gold call sequence. Tool execution uses a local deterministic mock registry (no live APIs), making every run exactly reproducible. Episode-level toolsets are drawn per episode; segments share the toolset subject to drift. Success predicate: a segment succeeds iff every gold call is emitted with the post-drift-correct schema (exact name and parameter match after drift rewrites) and no out-of-schema call is issued; segment SR is binary per segment, and episode metrics aggregate these binary outcomes.

3.2 Drift engine

Five drift families applied at segment boundaries (one event per boundary); see Table 1.

D1 is sampled at 2× weight. Drift targets are selected from the union of future segments' gold tools (§3.6), and D4 rewrites are only planned where the future gold actually uses a value different from the rewritten default—guaranteeing the rewrite changes behavior. Empirical drift timing: drift occurs after segment 1 in 77 episodes and after segment 3 in 85 episodes (of 100; some episodes have both), leaving segments 0–1 as the pre-drift baseline window. Family composition: D1 61, D2 36, D3 36, D5 28, D4 1 events—D4 is deliberately near-empty on the main pool (its planning constraint rarely fires; §3.6) and is powered up

separately in the D4-dense experiment (§4.3); the main results therefore make no claim about D4.

3.3 Protocol

- Drift is injected at segment boundaries; the environment (tool registry + docs) changes between segments; the agent continues the episode.
- Configuration axes: memory $\in$ {none, scratchpad} $\times$ notice $\in$ {off, on}. Scratchpad: a per-episode notes field the agent may write and read across segments. Notice: an explicit "the environment has been updated; re-read the tool documentation" message injected after each drift.
- Decoding: temperature 0, max 4 turns/segment, max 512 tokens; vLLM 0.6.6 (Qwen2.5 family) (Qwen Team, 2024), isolated vLLM 0.8.5 env for Qwen3-0.6B (Qwen Team, 2025); 8×L40.

3.4 Metrics

- Pre-drift SR: success rate before the first drift (capability baseline).
- Post-drift SR@k: success rate over the first k post-drift segments (immediate hit).
- Adaptation AUC: mean SR over post-drift segments, aggregated episode-weighted (each drift episode contributes its post-drift mean; run-level AUC is the mean over drift episodes). We emphasize an accounting identity: by the law of total expectation, run-level Adaptation AUC equals mean post-drift SR—by construction, not as an empirical finding. The trajectory-shape information therefore lives in the companion metrics (RS, recovery rate, OI), not in AUC; AUC's role is comparability across configurations, and the decomposition identity verifies the arithmetic of budget–quality closure, not causal mediation.
- Recovery Steps (RS): per drift episode, $\min\{k : \text{SR@}k \geq \max(0.9 \cdot \text{pre}, 0.5)\}$; run-level RS averages recovered episodes only, and non-recovered episodes are reported separately as the recovery rate (fraction of drift episodes with finite RS). Recovery rates range 0.39–0.74 across the 24 runs (e.g., 1.5B nomem 0.74, 0.5B mem 0.49, 3B-base nomem 0.42): most configurations recover in the majority of drift episodes, with substantial model spread.
- Oscillation Index (OI): std/mean of post-drift SR per episode, averaged over drift episodes (values 1.23–1.57 across the 24 runs; higher = more erratic recovery).
- Normalized Recovery (NR) = $\overline{\text{AUC}} \,/\, \overline{\text{pre-drift SR}}$ over drift episodes. Motivation: the absolute-AUC scale gradient narrows sharply under NR (nomem: 0.5B 0.856 > Qwen3-0.6B 0.789 > 1.5B 0.784); NR decouples "recovers well" from "was strong to begin with," and AUC/NR must be reported side by side.

3.5 Quality validators

- V1 gold executable (in the static environment).
- V2 snapshot/replay consistency (deterministic mock execution).
- V3 drift observable: a constructed stale call (pre-drift schema) must fail post-drift.
- V4 drift effective: every D1–D4 target tool must appear in future segments' gold.

3.6 Benchmark validity methodology (a core contribution)

Two validity flaws recur across drift-benchmark designs; our construction avoids both, and the shipped validators let future work check for them.

1. Drift–task misalignment. If drift targets are drawn from the current segment's gold while the rewrite applies to future segments, the future gold may never call the target tool—drift targets are selected from the union of future gold tools and pass a static V4 validator, so every drift event is observable and effective.

2. Position primacy. Merging toolsets by segment order places early-segment gold tools at the head of the list (hit rate 0.63 at rank 0–2 vs. 0.40 at rank 15+); we therefore sort toolsets alphabetically (mean rank 8.2–8.6 across positions) and report rank-bucketed hit rates as a fairness check, so post-drift comparisons are not confounded by list position.

3. Validation protocol (reusable). Treated-vs-control: hit rate on drifted target tools vs. undrifted tools in the same segments, with a z-test. Without future-gold target selection the difference is 0 ($z \approx 0$); with it, the difference is -0.23 ($z = -3.4$—drift roughly halves target-tool success).

4. Counterfactual pool construction. A no-drift control pool must restore both the gold sequence and the toolset snapshots; stripping only drift_events leaves "new-schema gold + old-schema tools"—an unsolvable task that systematically depresses the control. Control pools pass the same validator suite as drift pools and are checked for drift_events = 0.

5. Statistical test choice. Memory/notice effects are highly sparse (> 60% of episode diffs = 0); percentile bootstrap is anti-conservative here. We standardize on the two-sided sign-flip permutation test (20k, episode-level paired)—exact under H0, robust to sparsity—and always report sign counts (+/−/0) to expose sparsity.

6. Rare-drift-type sparsity (D4). D4 target selection scans all future-gold tools (name-sorted for determinism) and requires the future gold to actually use a value ≠ the rewritten default, so each D4 rewrite changes behavior (enum params occur in ∼3% of cases). The D4-dense pool (Appendix C) scales this to 71 events with a paired control on the identical rng stream.

3.7 Budget–quality decomposition (behavioral mediation of intervention effects)

For any prompt-level intervention I (memory, notice; extensible to RAG injection, system-prompt changes), the SR effect decomposes into two channels:

- Budget channel: does I change agent calls/segment? Segment-level paired comparison of n_agent_calls, partitioned into budget-cut / equal / expanded segments.
- Quality channel: at equal call counts, does SR change (gold coverage or SR@equal)? The two proxies cross-validate (memory: Qwen3 cov@eq +0.030 / SR@eq +0.033; 0.5B −0.018/−0.023).
- Closure check: overall $\Delta\text{SR} = P(\text{cut})\cdot\Delta\text{SR}|_{\text{cut}} + P(\text{equal})\cdot\Delta\text{SR}|_{\text{equal}} + P(\text{expanded})\cdot\Delta\text{SR}|_{\text{expanded}}$. Both interventions × all models close exactly (residual < 0.001)—within the measured budget and quality channels, no residual third channel is detectable.
- Reporting rule: an intervention effect must always be reported with Δcalls/seg (behavioral effect) and $\text{dSR}|_{\text{equal}}$ (quality effect); an AUC difference alone conflates "changed behavior" with "changed quality."

4 Experiments

4.1 Setup

Models: Qwen2.5-{0.5B, 1.5B}-Instruct, Qwen3-0.6B, Qwen2.5-3B (base), Qwen2.5-Coder-3B-Instruct, Qwen2.5-7B (base) (Qwen Team, 2024; 2025). Engine: vLLM 0.6.6 (Qwen2.5 family); Qwen3-0.6B on an isolated vLLM 0.8.5.post1 + torch 2.6.0 env; temperature= 0, max_turns= 4, max_tokens= 512; 8×L40. Pools: a primary pool of 100 episodes × 6 segments, a no-drift counterfactual on the same stream, and a D4-dense paired experiment (dense vs. control, 71 events, 47 differing episodes). Statistics: two-sided sign-flip permutation, 20k permutations, episode-level pairing, sign counts (+/−/0) always reported.

Table 2: Main results. 100 episodes × 6 segments. RS = mean Recovery Steps. The bottom row is the no-drift counterfactual (post-window SR over seg2–5).

Model	Config	Mean SR	Pre-drift SR	Adapt. AUC	Rec. Steps
Qwen2.5-0.5B-Inst	none	0.222	0.233	0.207	2.30
Qwen2.5-0.5B-Inst	scratchpad	0.180	0.209	0.153	2.35
Qwen2.5-0.5B-Inst	none+notice	0.225	0.233	0.218	2.18
Qwen2.5-0.5B-Inst	scratchpad+notice	0.180	0.209	0.153	2.38
Qwen3-0.6B	none	0.255	0.286	0.231	1.87
Qwen3-0.6B	scratchpad	0.282	0.300	0.263	1.79
Qwen3-0.6B	none+notice	0.265	0.291	0.247	1.91
Qwen3-0.6B	scratchpad+notice	0.278	0.300	0.261	1.80
Qwen2.5-1.5B-Inst	none	0.345	0.390	0.312	2.01
Qwen2.5-1.5B-Inst	scratchpad	0.335	0.406	0.288	2.12
Qwen2.5-1.5B-Inst	none+notice	0.325	0.390	0.274	2.12
Qwen2.5-1.5B-Inst	scratchpad+notice	0.327	0.406	0.272	2.18
Qwen2.5-Coder-3B	none	0.258	0.298	0.237	2.26
Qwen2.5-Coder-3B	scratchpad	0.273	0.291	0.269	2.18
Qwen2.5-Coder-3B	none+notice	0.280	0.298	0.282	2.13
Qwen2.5-Coder-3B	scratchpad+notice	0.265	0.291	0.253	2.28
Qwen2.5-3B (base)	none	0.140	0.139	0.142	2.08
Qwen2.5-3B (base)	scratchpad	0.143	0.150	0.132	2.50
Qwen2.5-3B (base)	none+notice	0.137	0.139	0.137	2.36
Qwen2.5-3B (base)	scratchpad+notice	0.152	0.150	0.142	2.63
Qwen2.5-7B (base)	none	0.177	0.186	0.177	2.28
Qwen2.5-7B (base)	scratchpad	0.153	0.188	0.132	2.29
Qwen2.5-7B (base)	none+notice	0.177	0.186	0.172	2.46
Qwen2.5-7B (base)	scratchpad+notice	0.177	0.188	0.172	2.45
Qwen2.5-1.5B-Inst	none, no-drift counterfactual	0.375 (post-window SR 0.366)			

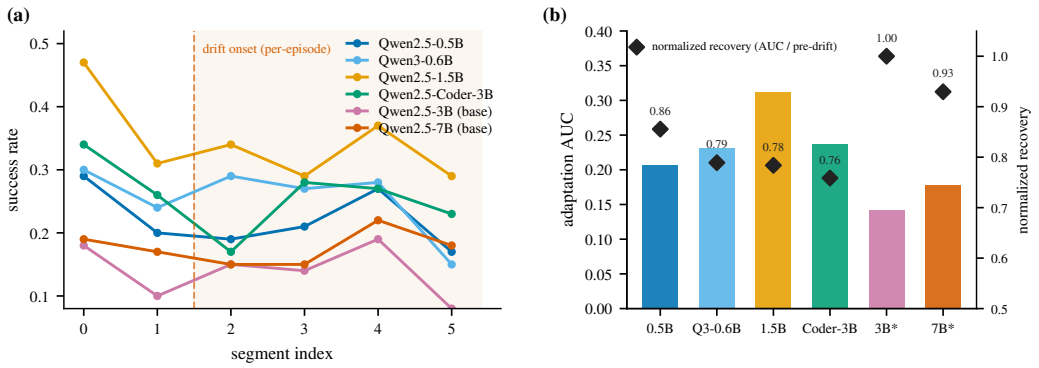


Figure 1: Main results (nomem runs). (a) Per-segment success rate; drift is injected at segment boundaries (per-episode timing), and the shaded region marks the typical post-drift window scored by Adaptation AUC. (b) Adaptation AUC by model (bars, left axis) alongside Normalized Recovery (AUC/pre-drift over drift episodes, diamonds, right axis): the absolute scale gradient largely flattens out under normalization, indicating the absolute gap is mostly baseline capability. Base models marked ∗.

4.2 Main results

4.3 Findings

F1. Drift impact is measurable. Post-drift hit rate on drifted target tools is 0.242 (66 calls) vs. 0.468 on undrifted tools in the same segments (299 calls; two-proportion $z = -3.36$, $p = 8 \times 10^{-4}$; episode-clustered $z_{\mathrm{CR1}} = -3.66$, 93 episodes): drift roughly halves target-tool success.

Table 3: Budget–quality decomposition of the memory effect (nomem−mem), post-drift segments. Closure column: predicted (observed).

model	Δcalls/seg	p	P(cut)	$\Delta SR\|_{cut}$	$\Delta SR\|_{=}$	$\Delta SR\|_{up}$	closure (obs.)
0.5B	+0.618	< 0.0001	37.8%	−0.093	−0.023	−0.028	−0.050 (−0.050)
1.5B	+0.220	< 0.0001	20.2%	−0.259	+0.021	+0.171	−0.022 (−0.023)
Qwen3-0.6B	+0.025	0.52	10.8%	+0.000	+0.033	+0.107	+0.035 (+0.035)
Coder-3B	−0.077	0.13	13.0%	−0.096	+0.000	+0.212	+0.023 (+0.023)

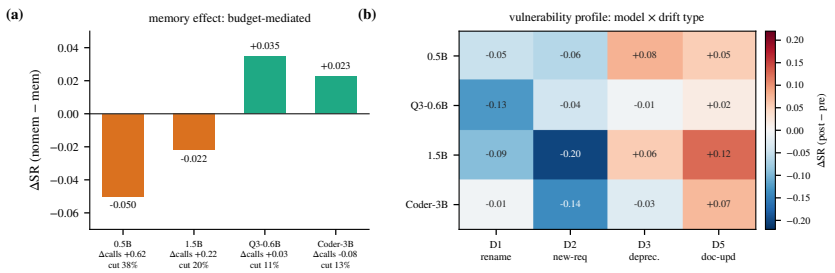


Figure 2: Mechanisms. (a) Budget–quality decomposition of the memory effect: harm on 0.5B/1.5B coincides with call-budget cuts (Δcalls > 0 means memory reduces calls), while benefits on Qwen3-0.6B/Coder-3B come from equal-call quality gains and budget expansion respectively. (b) Vulnerability profiles (post-drift minus pre-drift SR, by drift type, nomem): D1 hurts all models; D2 crushes 1.5B (−0.20) but barely touches Qwen3-0.6B; D3/D5 help some models (e.g., +0.12 for 1.5B under D5). Difficulty is a model×type interaction, not a property of the drift type alone.

F2. Scale gradient in adaptation—but flat normalized recovery. nomem AUC: 0.5B 0.207 < Qwen3-0.6B 0.231 < 1.5B 0.312; sign-flip Δ(1.5B−0.5B)= +0.105, CI= $[+0.056, +0.153]$, $p = 0.0001$. Yet NR is nearly flat (0.5B 0.856 > Qwen3-0.6B 0.789 > 1.5B 0.784): much of the absolute gap is baseline capability, not adaptation. The recovery-rate picture complements this: 1.5B recovers in 74% of drift episodes vs. 60% for 0.5B and 56% for Qwen3-0.6B—larger models both recover more often and plateau higher. Absolute and normalized scores must be reported together (Fig. 1b).

F3. Memory value flips sign across models. Scratchpad significantly hurts 0.5B (nomem−mem = +0.054, sign-flip $p = 0.0037$; signs +23/−8/0:62), direction-consistent slight harm on 1.5B (+0.024, $p = 0.244$), and helps Coder-3B (−0.032, $p = 0.011$); Qwen3-0.6B is direction-consistent but borderline (−0.032, $p = 0.058$). Applying Holm correction across the six per-model memory contrasts (the prespecified family; notice, scale, D4, and trend tests are reported uncorrected), 0.5B ($p_{Holm} = 0.022$), 7B-base ($p_{Holm} = 0.025$), and Coder-3B ($p_{Holm} = 0.043$) survive at α=0.05, and Qwen3-0.6B does not ($p_{Holm} = 0.17$)—we therefore describe it as suggestive, not significant. Effects are sparse (62/93 episode diffs = 0 on 0.5B). Pooling across models would average a harm and two benefits into a spurious "no effect"—memory value is a model×environment interaction, not a universal constant.

F3b. Mechanism: budget–quality decomposition closes exactly for all models. Table 3 and Fig. 2a show the decomposition. Harm and benefit both travel through behavior: on 0.5B/1.5B the harm is budget crowding (scratchpad text displaces calls; harm concentrates in budget-cut segments; quality at equal calls ≈ 0); Coder-3B's benefit is budget expansion (SR +0.212 in expanded segments); Qwen3-0.6B keeps its budget but chooses better at equal calls (+0.033).

F4. Drift-type × model vulnerability profiles. ΔSR (post-drift − pre-drift) by type, nomem runs (Fig. 2b): D1 hurts everyone (the most universal fragility, −0.05 to −0.13); D2 crushes 1.5B/Coder (−0.20/−0.14) but barely touches Qwen3 (−0.04); D3 helps Qwen3 (+0.06—it

Table 4: D4-dense paired experiment (dense − control, post-drift SR, n=47, sign-flip 20k).

model	ΔSR (dense−ctrl)	signs	p	D4 obs	new-value adoption
0.5B	−0.0213	+0/−4/0:43	0.12 ns	78	1
1.5B	**−0.0851**	+0/−15/0:32	**< 0.0001**	62	2
Qwen3-0.6B	**−0.0585**	+0/−11/0:36	**0.0013**	69	2
Coder-3B	−0.0479	+0/−9/0:38	**0.0043**	72	0

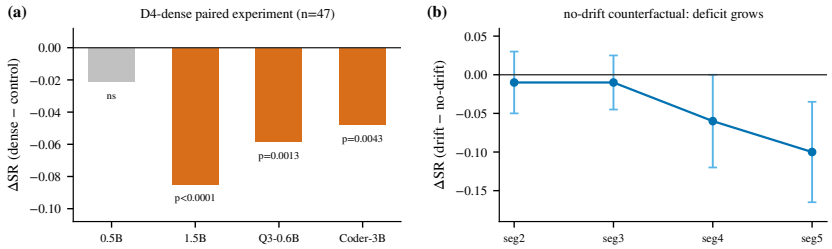


Figure 3: Causal lenses on drift damage. (a) D4-dense paired experiment (n=47): post-drift SR loss under dense no-error-signal drift (D4) vs. paired control; gray = ns. (b) No-drift counterfactual: the SR deficit grows from seg2 (−0.01) to seg5 (−0.10)—failure is cumulative. Error bars: 95% bootstrap CIs.

triggers more careful doc reading); D5 helps 1.5B (+0.12—a "benign drift" that advertises correct usage). Difficulty is not a property of the drift type alone: it is a model×type interaction, and the benchmark emits per-model vulnerability profiles.

F4b. D4—drift without an error signal—is a systematic blind spot. New-enum-value adoption was 0/176 across 4 models: the old value remains schema-valid, so nothing ever errors—the agent must read the documentation to notice anything at all. We therefore built a D4-dense paired experiment: rebuilding the pool on the identical rng stream with D4-first planning yields 71 D4 events (5.5×) with only 47/100 episodes changed; the paired control drops the event exactly where D4 is plannable (construction details in Appendix C).

Paired result (dense − control, post-drift SR, n=47, sign-flip 20k; Table 4, Fig. 3a): pre-drift pairing is exact (0/47 nonzero per-model seg0–1 differences). New-value adoption is 5/281 = 1.8% (four of the five from a single episode—isolated cases, not a strategy). The harm concentrates in the first post-drift segment (1.5B −0.106, Qwen3 −0.128 at seg2) and persists to seg5 with no recovery—with no error signal, nothing triggers correction. D4's no-error-signal mechanism is orthogonal to D1/D2/D3 (schema-level hard failures): even D2, whose failure is at least visible, is hit less. This is demonstrated on this corpus, whose hermes-derived tasks offer only 14 usable (tool, enum-param) pairs with ≥ 2 distinct gold values; broader corpora may behave differently.

F5. Drift notices backfire—conditionally, and the harm is budget-linked. On 1.5B nomem the notice reduces adaptation (ΔAUC = +0.038, CI= $[+0.013, +0.065]$, sign-flip $p = 0.0088$; signs +16/−4/0:73). The harm is budget-linked: notice cuts calls 2.00 → 1.75/segment ($\Delta = -0.25$, CI= $[-0.34, -0.16]$); the SR loss concentrates in the 16.5% of segments whose budget shrank (SR 0.409 → 0.182), while gold coverage at equal call counts is unchanged (0.424 vs. 0.422)—quality effect ≈ 0; closure: $0.165 \times (-0.227) + \sim 0 \approx -0.038$. On 0.5B and Qwen3-0.6B it changes nothing. On Coder-3B nomem the notice helps (−0.046, $p = 0.0058$; $p_{\text{Holm}} = 0.035$) by the same channel in reverse (more calls; SR +0.340 in the 11.8% expanded segments). On 7B base mem the notice rescues the scratchpad harm (−0.040, $p = 0.021$): budget expansion restores calls to the nomem level (0.132 → 0.172 vs. 0.177). The effect flips sign across models×conditions; within the Qwen family at ≤7B, one pattern holds: an intervention only has an effect where it changes behavior. Robustness checks: raising the cap to *max_turns*=6 leaves the 0.5B memory harm intact (−0.073 vs. −0.054; n.s. difference, sign-flip p=0.51; 32.5% of memory-arm segments still hit the cap), ruling out

cap displacement; the notice harm is unchanged (-0.038). A context-reset arm (scratchpad cleared after drift) recovers 0.5B mem ($0.153 \rightarrow 0.194$) while the decisive reset-only control is unchanged ($0.207 \rightarrow 0.210$): memory harm shrinks $+0.054 \rightarrow +0.016$, attributing the recovery to removing stale memory content, not the reset procedure.

F5'. Causal drift effect via no-drift counterfactual. Paired against a rebuilt no-drift pool (drift events stripped AND pre-drift gold/toolsets restored), drift causally reduces post-window SR (segments 2–5) by -0.048 (sign-flip $p = 0.0008$; signs $+3/-22/0{:}68$), and the deficit grows with exposure: seg2 $-0.01 \rightarrow$ seg4 $-0.06 \rightarrow$ seg5 -0.10 ($p = 0.002$; Fig. 3b)—failure is cumulative.

F6. Instruction tuning dominates scale. 7B-base (AUC 0.177) and 3B-base (0.142) fall far below 1.5B-Instruct (0.312): $\Delta(1.5\text{B}-7\text{B}) = +0.134$, $p = 0.0001$; $\Delta(1.5\text{B}-3\text{B}) = +0.169$, $p < 0.0001$. 0.5B-Instruct outperforms 6×-larger 3B-base ($+0.065$, $p = 0.021$); Coder-3B underperforms 1.5B-Instruct despite 2× params ($p = 0.004$)—domain mismatch. Base-model inertness is conditional: 7B-base memory harms ($+0.046$, $p = 0.005$, budget-linked) while 3B-base stays inert ($p = 0.71$).

F6'. Robustness $\neq$ adaptation. Within every model run (9 runs), episode-level pre-drift SR vs. adaptation AUC correlate at $|r| \leq 0.16$ (median 0.09)—static capability does not predict recovery; adaptation is episode-specific (Appendix C).

5 Discussion

The double edge of memory. Scratchpad can amplify stale patterns in non-stationary environments, but its value flips sign by model: harm = budget crowding (0.5B: -0.62 calls/segment), benefit = budget expansion (Coder-3B: $+0.212$ SR in expanded segments) or better selection at equal calls (Qwen3: $+0.033$). Three causal lenses converge: the no-drift counterfactual (-0.048, $p = 0.0008$, growing to -0.10 by seg5), the D4-dense no-error-signal mechanism (-0.085 on 1.5B, no recovery), and the treated-vs-control contrast (0.242 vs. 0.468).

Behavioral mediation as a unified framework. Both interventions act through the same two channels (budget, quality) and both decompositions close exactly: the sign of an intervention is set by where behavior changes, not by information content—pooling across models would cancel a harm and two benefits into a spurious "no effect" (F3). Limitations: deterministic mock execution, single domain, episode length 6, scale $\leq$7B, one scratchpad design, notice wording/timing unexplored. Extending the same drift-injection protocol and validity checks to live-API environments, longer horizons, and larger models is a natural next step.

6 Conclusion

AgentDriftBench makes "whether and how agents recover from drift" measurable, decomposable, and statistically testable; we will release the drift engine, validators, pools, and analysis scripts publicly upon acceptance, to make adaptation dynamics a first-class measurement target.

Reproducibility statement

Pool builders (build_task_pool.py, build_task_pool_v4dense.py; seed=42, rng-stream-preserving), the evaluation harness (run_eval.py), all analysis scripts (metrics, unified sign-flip protocol, budget–quality decomposition, counterfactual and paired-D4 analyses), and all 64 run directories are included with the code. Code, prompts, and data will be released publicly upon acceptance. Pools are reproducible from the tracked builders; the D4-dense pairing is verifiable by the 47/100 episode-diff check.

Ethics statement

This benchmark evaluates agents in a fully synthetic, locally-executed environment; no live APIs, user data, or human subjects are involved. Drift injections are simulated transformations of public tool-calling corpora. We release the benchmark to encourage robustness measurement; adversarial misuse potential is limited by the deterministic mock registry and published task pools.

AI use statement

Generative AI tools were used for drafting and editing parts of the manuscript text, generating plotting code for the figures, and assisting with literature search. All benchmark design, experiments, data analysis, and statistical tests were designed and executed by the authors, who verified all AI-assisted outputs (every printed number was recomputed from archived run records; every citation was verified against the source). No AI system was used to generate or label evaluation data: the task pools, drift injections, and mock tool registry are deterministic and built from public corpora by tracked scripts. The authors take full responsibility for the final content of this work.

References

Mohammad Hasbi Assidiqi, Daniyal Alghazzawi, Suaad Alarifi, and Cheng Li. Benchmarking reference-free LLM agent robustness under schema, policy, and toolset drift. IEEE Access, 14:79662–79672, 2026. doi: 10.1109/ACCESS.2026.3696096.

Victor Barrès, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. arXiv preprint arXiv:2506.07982, 2025.

Tongbo Chen, Zhengxi Lu, Zhan Xu, Guocheng Shao, et al. KnowU-Bench: Towards interactive, proactive, and personalized mobile agent evaluation. arXiv preprint arXiv:2604.08455, 2026.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. ACM Computing Surveys, 46(4):1–37, 2014.

Zhicheng Guo et al. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In Findings of ACL, 2024. arXiv:2403.07714.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, et al. WILDS: A benchmark of in-the-wild distribution shifts. In International Conference on Machine Learning (ICML), 2021. arXiv:2012.07421.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Wang, Qixiang Zhang, Yanan Liu, Daya Guo, et al. AgentBench: Evaluating LLMs as agents. In International Conference on Learning Representations (ICLR), 2024. arXiv:2308.03688.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. arXiv preprint arXiv:2311.12983, 2023.

Nous Research. hermes-function-calling-v1 dataset. https://huggingface.co/datasets/NousResearch/hermes-function-calling-v1, 2024.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, et al. The berkeley function calling leaderboard (BFCL). Berkeley Gorilla Project, https://gorilla.cs.berkeley.edu/leaderboard.html, 2025. Public leaderboard; accessed September 2026.

Qwen Team. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024.

Qwen Team. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, et al. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. arXiv preprint arXiv:2508.20453, 2025.

Weiwei Xie, Shaoxiong Guo, Fan Zhang, Tian Xia, et al. MemEvoBench: Benchmarking safety risks from memory misevolution in LLM agents. arXiv preprint arXiv:2604.15774, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. arXiv preprint arXiv:2406.12045, 2024.

Junjie Ye et al. RoTBench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. In Proceedings of EMNLP, 2024. arXiv:2401.08326.

Yuhao Zhan, Bingxiang He, Zecong Tang, and Chaojun Xiao. PACE-Bench: Benchmarking physics adaptation via code evolution in dynamic environments. arXiv preprint arXiv:2608.14441, 2026.

Junhao Zheng et al. LifelongAgentBench: Evaluating LLM agents as lifelong learners. arXiv preprint arXiv:2505.11942, 2025.

A Prompt templates and decoding details

The agent system prompt specifies the tool-calling contract; the scratchpad variant appends a per-episode notes field (read/write across segments); the notice variant injects "The environment has been updated; re-read the tool documentation" after each drift event. Decoding: temperature 0, max 4 agent turns per segment, max 512 completion tokens per turn. Qwen2.5-family models run on vLLM 0.6.6; Qwen3-0.6B runs on an isolated vLLM 0.8.5.post1 + torch 2.6.0 environment (engine compatibility). All executions hit a deterministic local mock registry.

B Drift-type statistics per pool

Drift composition on the main pool: D1 61, D2 36, D3 36, D5 28, D4 1 events (D4 is deliberately near-empty on the main pool; the D4-dense experiment scales it to 71). The dense/control pair differs in exactly 47/100 episodes; all gold diffs are explained by D4 param sets, and tool diffs are D4 doc-appends only.

C D4-dense pool: design notes

The D4 finding required powering up a sparse drift family without breaking pairing. Injection into empty boundaries was impossible (0/26 plannable—boundaries are empty because D4 cannot plan there; the corpus has only 14 (tool, enum-param) pairs with ≥ 2 distinct gold values, of which 12 appear in the main pool). The dense pool therefore rebuilds on the identical seed/rng stream, consuming rng.choice unconditionally before overriding the drift type to D4—this consume-then-override pattern preserves episode pairing (47/100 episodes differ) and is a reusable technique for any conditional planner layered onto a stochastic stream. The control pool drops the event exactly where D4 is plannable.