DYNAMIC WRITES, SIMPLE SCHEDULES: AN ORACLE STUDY OF IN-PLACE TEST-TIME TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time training (TTT) adapts a language model as it processes an input, but existing methods typically apply a fixed adaptation configuration throughout the context. We examine whether varying where and when the model writes can outperform the best constant write action. Using a fixed Qwen3-1.7B In-Place TTT checkpoint, we compare the per-example best constant action against a bounded oracle search over chunkwise dynamic trajectories. Across 6,500 examples from 13 RULER-16K tasks, the best trajectories returned by the bounded oracle search increase the mean score from 78.38 to 79.77, a gain of 1.39 percentage points. Among the 1,933 non-ceiling examples, the bounded oracle search improves 170 and ties the remaining 1,763, increasing their mean score from 27.29 to 31.96 ($+4.67$ points). Controlled interventions show that successful outcomes depend on both the ordering of write actions and the model state accumulated across chunks. Nevertheless, complex switching is often unnecessary: 62 of 78 improved multi-switch trajectories admit an exactly score-matched one-switch alternative. We observe similar, though smaller, gains on Qwen3-4B and Llama-3.1-8B under their respective 32K-token protocols, improving over the per-example best constant action by 0.79 and 0.36 points, respectively. These results establish an oracle-reachable advantage for dynamic in-place TTT and suggest that relatively simple switching policies may capture much of this opportunity. It motivates learning policies that select write actions from the evolving context and adaptation state.

1 INTRODUCTION

Test-time training (TTT) equips a language model with a temporary, input-dependent state that is updated as the model processes its context. Prior work has instantiated this state through fast-weight (Ba et al., 2016; Schlag et al., 2021), in-place parameter updates (Feng et al., 2026), predictive objectives (Ouyang et al., 2026), recurrent inner learners (Sun et al., 2024), and end-to-end long-context training (Tandon et al., 2025). Although these approaches differ in how the temporary state is represented and optimized, they typically apply a fixed adaptation configuration across the input and update a predetermined set of parameters.

Related work suggests that adaptation can instead be selective along two complementary axes. In test-time adaptation, EATA and SAR select which test examples contribute to model updates (Niu et al., 2022; 2023), whereas TENT and EcoTTA restrict adaptation to predefined parameter subsets or lightweight modules (Wang et al., 2021; Song et al., 2023). Parameter-efficient transfer learning similarly limits updates to adapters, bias terms, or low-rank parameterizations (Houlsby et al., 2019; Ben Zaken et al., 2022; Hu et al., 2022). However, these methods generally do not vary the selected parameter group over the course of a single input. Such uniformity may be restrictive for long contexts containing segments that differ in content, structure, and function. We therefore study dynamic TTT policies that vary both when adaptation is performed and which parameter groups are updated across input chunks.

In this work, we investigate whether TTT benefits from varying its adaptation configuration within an input. We represent chunkwise adaptation as a trajectory of *write actions*, where each action specifies which parameter group is updated on a given chunk. A *static* trajectory applies the same action to every complete chunk, whereas a *dynamic* trajectory may change actions as the context and

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.

temporary model state evolve. For each example, we define the Constant Oracle (CO) as the highest-scoring static trajectory for each example and ask a controlled question: can any non-constant trajectory outperform the best constant action for that example?

To answer this question, we evaluate all seven constant-action trajectories and compare with the per-example CO. We then perform a bounded coordinate search over non-constant trajectories to select the best observed candidate. Constant trajectories remain in the candidate pool throughout the search, ensuring that each selected trajectory is compared against the CO under the same checkpoint, action space, and execution protocol. We refer to the resulting improvement as *sequential utility*: the additional performance attainable by changing write actions within an example after accounting for the best constant action.

In Figure 1, we intervene on action order and accumulated fast weights to examine how the write schedule and carried state affect continuation. We then test whether matching boundary fast weights is sufficient when prefix-cache histories differ, and whether reproducing the observed scores requires complex switching.

Our main contributions are:

- We formulate chunkwise in-place TTT as a trajectory-design problem and quantify the additional utility beyond the per-example best constant action through bounded, answer-aware search. The 13-task evaluation on Qwen3-1.7B yields a 1.39-percentage-point gain over the per-example best constant action.
- We use schedule and state interventions to show that continuation is sensitive to the organization of write actions and the accumulated fast-weight state. Prefix-cache replacement further changes continuation even when boundary fast weights and the suffix execution path are matched.
- We distinguish history dependence from switching complexity: exhaustive One-Switch Oracle (OSO) evaluation identifies exactly score-matched one-switch alternatives for 62 of 78 improved multi-switch trajectories.

Taken together, our results suggest that test-time training should be studied not only through the design of its update rule, but also through the design of its adaptation trajectory. This perspective jointly considers *what* to update, *when* to update it, and *what state* each update leaves for subsequent computation. It motivates learning policies that condition write decisions on the evolving context and temporary model state.

2 Static Actions v.s. Dynamic Actions

A static write protocol applies the same action to every complete chunk, providing a strong per-example reference while fixing the write location throughout the context. We define the chunkwise evaluation protocol and static and dynamic trajectories, then compare them on the 13-task evaluation set. This tests whether within-example action changes improve task score over the best constant action and establishes the sequential score gap examined by subsequent mechanism analyses.

2.1 Experimental Setup

The main mechanism analyses use a fixed continued-trained Qwen3-1.7B checkpoint (Yang et al., 2025); Table 1 additionally reports Qwen3-4B (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024). All model-scale comparisons use the 13 RULER-16K tasks (Hsieh et al., 2024). For the Qwen3-1.7B mechanism analysis, we use a 16K-token context, a chunk size of $C = 1024$, and writable MLP down-projection layers $\mathcal{L}_{\mathrm{TTT}} = \{0, 6, 12, 18, 24\}$. The checkpoint is frozen throughout evaluation, and every example starts from the same checkpoint state. An input x is divided into T_x complete chunks, and one action is assigned to each chunk (For Qwen3-4B and Llama-3.1-8B we use 32K-token context).Full model-specific configurations, the Base reference, and additional update details are provided in Appendix A.

The action space is $\mathcal{A} = \{\textsc{Off}, L_0, L_6, L_{12}, L_{18}, L_{24}, \textsc{All}\}$. Off suppresses the new write for the current chunk, L_k writes only to layer k, and All writes to all five writable layers. For a trajectory

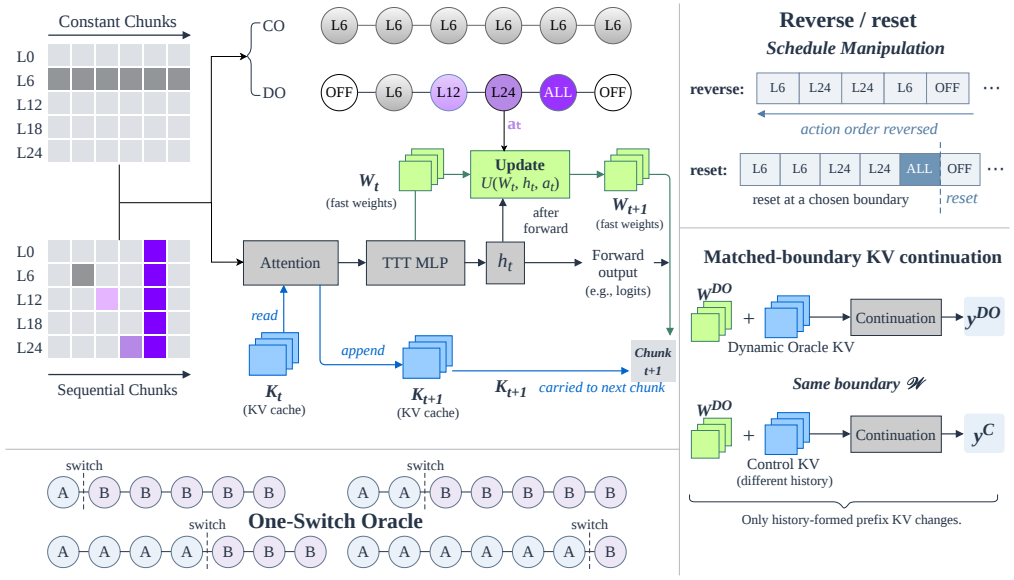


Figure 1: Overview of dynamic write trajectories and the diagnostic experiments. **Left:** Static trajectories repeat one write action, whereas dynamic trajectories vary actions across chunks. Under a fixed execution path, writes update the fast weights for subsequent chunks, while attention accumulates a key/value cache. **Upper right:** Reverse changes the assignment of actions to input chunks while preserving their counts; Reset restores checkpoint fast weights at a suffix boundary. **Lower right:** The cache intervention compares Dynamic Oracle (DO) and control prefix KV histories with identical fast weights at the boundary and the same subsequent action schedule. **Bottom:** Exact One-Switch Oracle enumerates trajectories with two distinct constant actions and exactly one switch, carrying accumulated state across the switch.

$\pi_x = (a_1, \ldots, a_{T_x})$, let $W_{\ell,t}$ denote the effective weight at layer ℓ when chunk t is processed. The update computed from chunk t is

$$\Delta W_{\ell,t} = \mathrm{clip}_{F,\epsilon}\left(\eta \sum_{j=1}^{C-1} v_{\ell,t,j} u_{\ell,t,j}^{\top}\right), \qquad \eta = 1, \quad \epsilon = 10^{-5}, \tag{1}$$

where $u_{\ell,t,j}$ is the incoming feature and $v_{\ell,t,j}$ is the gated next-position target. The target projection and signed channel gate are fixed during evaluation.

Each chunk is first evaluated using the state available at its entrance. Its update is applied afterward and can therefore affect only later chunks. The current chunk does not use its own newly computed update. The incomplete input tail and generated tokens read the accumulated fast-weight state but do not perform additional writes. The action-dependent write mask is

$$m_\ell(\textsc{Off}) = 0, \qquad m_\ell(L_k) = \mathbf{1}[\ell = k], \qquad m_\ell(\textsc{All}) = 1, \tag{2}$$

so that

$$W_{\ell,t+1} = W_{\ell,t} + m_\ell(a_t)\Delta W_{\ell,t}. \tag{3}$$

Thus, OFF disables the current new write while retaining any state accumulated earlier in the trajectory.

A static trajectory repeats one action:

$$\pi_x^{\mathrm{static}} = a^{T_x}, \qquad a \in \mathcal{A}.$$

For the task score $R(x, \pi) \in [0, 1]$, we define the per-example static reference

$$R_{\mathrm{CO}}(x) = \max_{a \in \mathcal{A}} R(x, a^{T_x}). \tag{4}$$

	Qwen3-1.7B (16K)			Qwen3-4B (32K)			Llama-3.1-8B (32K)		
Protocol	**All**	**Non-ceil.**	**Imp.**†	**All**	**Non-ceil.**	**Imp.**†	**All**	**Non-ceil.**	**Imp.**†
N	6,500	1,933	170	6,500	1,778	85	6,500	872	39
OFF	77.29	26.56	38.81	82.29	37.75	36.61	91.58	40.13	39.23
BASE	74.51	28.13	50.92	82.39	46.93	60.84	89.85	50.65	72.52
ALL	77.20	26.54	39.56	82.21	37.49	36.61	91.64	40.25	40.77
CO	78.38	27.29	40.41	83.23	38.71	37.14	92.06	40.83	40.77
DO-LB	**79.77**	**31.96**	**93.53**	**84.02**	**41.57**	**97.02**	**92.42**	**43.47**	**100.00**
Δ_{pp}	+1.39	+4.67	+53.12	+0.79	+2.86	+59.88	+0.36	+2.64	+59.23

Table 1: Evaluation results on RULER. Average accuracy (%) is reported, with the best results in **bold**. $\Delta_{\text{pp}} = \text{DO-LB} - \text{CO}$. NON-CEIL. denotes samples with CO $< 100\%$, while †IMP. denotes samples with DO-LB $>$ CO.

We report $100R$ as the task score. The CO is therefore the highest-scoring constant trajectory for each example, obtained by evaluating all seven constant actions.

The dynamic search allows the action to change across chunks. It starts from the seven constant trajectories and performs one forward coordinate sweep with beam width four. At each chunk position, all seven actions are evaluated for each active beam, and the affected suffix is replayed from the corresponding prefix state. The constant trajectories remain in the candidate pool, so the best returned trajectory cannot score below the CO. We refer to the score of the best retained candidate as the Dynamic Oracle Lower Bound (DO-LB): it is an observed lower bound for the bounded search, rather than an optimum over the full 7^{T_x} trajectory space. The final task score is used only after generation to score and select candidates; it is not provided to the model during the forward computation. The released search procedure gives a per-example upper bound of $24T_x + 25$ unique candidate evaluations.Full search, tie-breaking, replay, and candidate-accounting details are given in Appendix B, with execution-validity checks in Appendix G.

As shown in Figure 1, the comparison uses the complete 13-task scope with $N = 6{,}500$ examples. When $R_{\text{CO}}(x) = 1$, this example is a ceiling example. There are $4{,}567$ such examples; they remain included in the overall evaluation and contribute zero incremental gain because the static reference has already reached the maximum score. The remaining $1{,}933$ non-ceiling examples are searched directly, before inspecting the dynamic outcomes. Population confidence intervals are computed from paired, task-stratified bootstrap resampling of the per-example scores.

2.2 RESULTS

As shown in Table 1, across all 6,500 examples, DO-LB reaches a mean score of 79.7659, compared with 78.3767 for CO. The paired gain is 1.3892 percentage points, with a 95% confidence interval of $[1.1549, 1.6369]$ percentage points. This result shows that the evaluated dynamic trajectory family contains scores beyond the strongest constant action under the same checkpoint, scorer, and execution protocol.

The searched non-ceiling subset gives the more direct estimate of the available improvement when the static reference leaves room for additional score. Its mean score increases from 27.2883 to 31.9598, corresponding to a gain of 4.6715 percentage points with a 95% confidence interval of $[3.9110, 5.4690]$ percentage points. The smaller gain over all 6,500 examples reflects the inclusion of 4,567 ceiling examples, which contribute zero incremental gain.

At the per-example level, dynamic search strictly improves 170 of the $1{,}933$ searched examples, ties $1{,}763$, and worsens none. Thus, 8.79% of the searched examples improve, corresponding to 2.62% of the complete 13-task denominator. The 170 improved examples have a mean score of 40.4118 under CO and 93.5294 under the selected dynamic trajectory. The resulting 53.1176-point difference is useful for describing the conditional intervention cohort, but it is not a population estimate and should not replace the 4.6715-point searched-set gain.

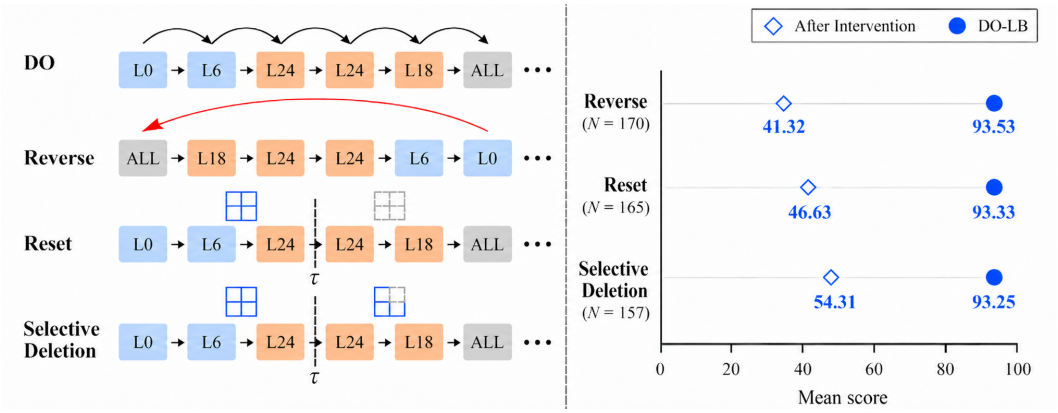


Figure 2: **Schedule and fast-weight interventions.** Left: Dynamic Oracle, Reverse, Reset, and Selective Deletion trajectories. Right: Paired average accuracy for each intervention and the corresponding Dynamic Oracle branch.

Because the constant trajectories remain in the candidate pool, the absence of negative dynamic differences is guaranteed by the search protocol. The result therefore establishes an observed score gap beyond the static action family, but it does not establish that the search is globally optimal or that it defines an answer-free online controller. An equal-budget random-search control is reported in Appendix B; under the matched trajectory-evaluation budget, DO-LB remains 3.57 percentage points higher. The next section examines whether this gap is associated with the order of actions and with the fast-weight state carried between chunks.

3 Schedule and Fast-Weight Interventions

Section 2 establishes that changing the write action within an example can improve over the strongest constant trajectory. We next examine what determines this improvement. Reverse tests whether the same action multiset remains effective when its assignments to input chunks are reversed. Reset and Selective Deletion instead modify the fast-weight state carried into a later suffix while preserving the corresponding continuation protocol. These interventions address different aspects of trajectory history and are analyzed separately before drawing a common conclusion.

3.1 Experimental Setup

Reverse. Let the selected trajectory for example x be $\pi_x^{\mathrm{DO}} = (a_1^{\mathrm{DO}}, \ldots, a_{T_x}^{\mathrm{DO}})$. The Reverse intervention constructs $\pi_x^{\mathrm{R}} = (a_{T_x}^{\mathrm{DO}}, \ldots, a_1^{\mathrm{DO}})$ and recomputes the complete trajectory from the common checkpoint. Reverse preserves the input and its token order, the trajectory length, the action identities and multiplicities, and the union of writable layers. It changes which action is assigned to each content chunk and the state in which later writes are executed. All counterfactual updates, hidden states, cache entries, and predictions are recomputed under the reversed trajectory.

Reverse is evaluated on the $N = 170$ examples for which the selected dynamic trajectory strictly exceeds CO. The comparison is not a content-independent permutation test: changing the schedule also changes the state passed to subsequent chunks. It therefore probes the combined effect of action–content alignment and state evolution.

Reset and Selective Deletion. For a Dynamic trajectory, let b_x denote the start of the final contiguous suffix selected by a frozen structural eligibility rule. The rule is defined from the trajectory before inspecting intervention outcomes and selects $N = 165$ examples. The exact eligibility criterion and boundary construction are detailed in Appendix D. State Reset replaces all writable fast weights immediately before this suffix with their checkpoint values:

$$W_{\ell,b_x^-}^{\mathrm{Reset}} = W_\ell^{\mathrm{ckpt}}, \qquad \ell \in \mathcal{L}_{\mathrm{TTT}}.$$

The native input, prefix cache, suffix actions, decoding configuration, and checkpoint are retained. The suffix is then recomputed from the reset state, including all later activations, fast-weight updates, cache entries, and predictions.

Selective Deletion removes the accumulated offset of one prescribed writable layer at the same boundary:

$$W^{\mathrm{Del}}_{\ell_x, b_x^-} = W^{\mathrm{DO}}_{\ell_x, b_x^-} - \left(W^{\mathrm{DO}}_{\ell_x, b_x^-} - W^{\mathrm{ckpt}}_{\ell_x} \right),$$

where ℓ_x is selected by a frozen prefix rule. The other boundary weights, the DO prefix cache, and the suffix actions remain unchanged, and the continuation is recomputed after the replacement. If the prefix contains an ALL action, each writable layer is evaluated in a separate branch and the resulting scores are averaged within the example. This defines a cohort of $N = 157$ examples for the Selective Deletion analysis.

3.2 Results and Analysis

Figure 2 summarizes the paired score comparisons. Exact Reverse produces a paired drop of $52.2059\,\mathrm{pp}$ (95% CI $[50.0000, 54.2941]$), with decreases, ties, and increases on $164/6/0$ examples. The reversed trajectory therefore returns close to the same-cohort CO reference, despite preserving the same action identities and multiplicities. The action multiset alone is insufficient to preserve the DO outcome. This result supports schedule–content alignment and history dependence, but does not isolate a content-independent ordering effect or establish algebraic non-commutativity of the update operator.

State Reset produces a paired drop of $46.7071\,\mathrm{pp}$ (95% CI $[43.3535, 49.9091]$) on the $N = 165$ cohort, with decreases, ties, and increases on $142/23/0$ examples. Replacing the accumulated fast-weight state before a fixed suffix therefore substantially changes the continuation.

Selective Deletion produces a paired drop of $38.9406\,\mathrm{pp}$ (95% CI $[35.1677, 42.5799]$) on the $N = 157$ cohort, with decreases, ties, and increases on $135/22/0$ examples. The result shows sensitivity to the prescribed layer-level state replacement. A magnitude-matched nonzero perturbation control reported in Appendix D scores $4.5418\,\mathrm{pp}$ higher than Selective Deletion (95% CI $[1.6709, 7.3474]$), indicating that perturbation magnitude alone does not explain the observed deletion effect.

Together, these interventions show that action order and accumulated fast-weight state have functional consequences on the discovered trajectories. Reverse mixes schedule–content alignment with state evolution, whereas Reset and Selective Deletion directly alter the carried fast-weight state. These effects are complementary rather than additive or causally decomposable. Cohort construction and cross-model replications of these interventions are reported in Appendices C and D.

4 Continuation Depends on Prefix KV History

The preceding interventions show that the fast-weight state carried into a suffix affects the final task score. So we next ask whether matching this state at the intervention boundary is sufficient to recover the DO continuation. As summarized in Figure 3, the two branches receive the same DO fast weights at the intervention boundary and share the suffix actions, decoding configuration, and execution path, but use prefix caches formed by different pre-boundary histories.

4.1 Experimental Setup

Let $\pi_x^{\mathrm{DO}} = (a_1^{\mathrm{DO}}, \ldots, a_{T_x}^{\mathrm{DO}})$ denote the selected trajectory for example x, and let b_x be the prespecified intervention boundary. We use the same frozen eligibility rule for all examples and obtain $N = 165$ paired trajectories.

We construct a control-history trajectory by replacing the DO prefix actions with OFF while retaining the DO suffix:

$$\pi_x^C = (\mathrm{OFF}, \ldots, \mathrm{OFF}, a_{b_x}^{\mathrm{DO}}, \ldots, a_{T_x}^{\mathrm{DO}}). \tag{5}$$

The control trajectory is used only to form a prefix-cache donor; it is not treated as an additional full-sample baseline.

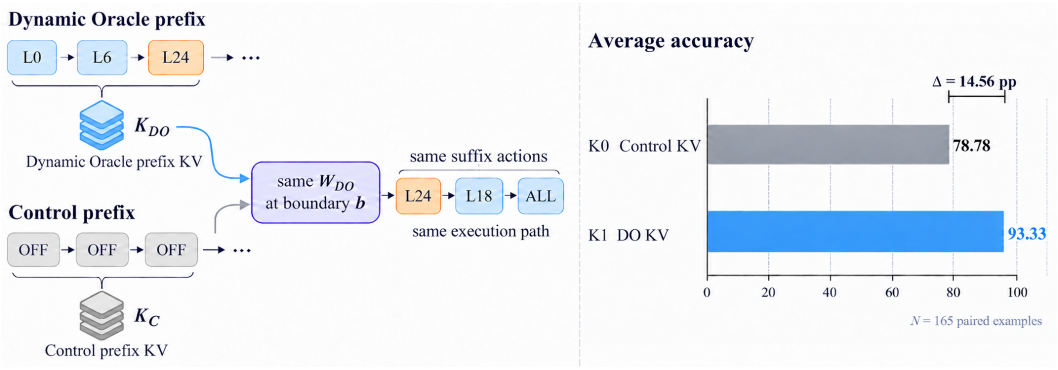


Figure 3: **Prefix-cache continuation intervention.** Left: Matched boundary fast weights, the same suffix action schedule, and a unified functional execution path; downstream states are recomputed after cache injection. Right: Average accuracy under the control prefix KV and DO prefix KV branches.

Let K_x^{DO} and K_x^C denote the complete prefix attention key/value caches produced by the DO and control histories, respectively. At the intervention boundary, both branches receive the same DO effective fast weights:

$$W^{(0)}_{x,b_x^-} = W^{(1)}_{x,b_x^-} = W^{\mathrm{DO}}_{x,b_x^-}. \tag{6}$$

We then compare

$$\mathcal{K}_0 = \left(W^{\mathrm{DO}}_{x,b_x^-}, K_x^C\right), \qquad \mathcal{K}_1 = \left(W^{\mathrm{DO}}_{x,b_x^-}, K_x^{\mathrm{DO}}\right). \tag{7}$$

Here K contains both keys and values at all attention layers. Only cache positions before b_x are taken from the selected donor.

Both branches use the same unified functional execution path for the prompt, the incomplete input tail, and generation. Consequently, the intervention changes the history represented by the prefix cache while leaving the numerical execution route unchanged. The suffix actions, input tokens, checkpoint, and decoding settings are also shared. After cache injection, all later hidden states, cache entries, fast-weight updates, tail computation, and generated tokens are recomputed from the injected boundary state. No future cache entry or DO update is copied into the control branch.

The equality of the fast weights is therefore a boundary condition. It does not imply that the two branches remain weight-identical during the suffix: different prefix caches can produce different activations and consequently different later fast-weight updates.

4.2 Results

As shown in Figure 3, replacing the DO prefix cache with the control-history cache reduces the mean task score from 93.33 to 78.78, corresponding to a paired drop of 14.56 percentage points (95% CI $[10.54, 18.77]$). Scores decrease, tie, and increase on $47/118/0$ examples, respectively. Because the two branches share the same boundary fast weights, suffix actions, and unified execution path, this gap shows that the boundary fast-weight state alone does not determine the continuation.

The contrast is a total downstream effect of replacing the starting prefix cache: subsequent activations, cache entries, and fast-weight updates are recomputed rather than held fixed. It therefore does not identify an isolated direct cache effect, establish the prefix cache as an independent learned memory, or show that $(W, K/V)$ is a universal sufficient state. The supported conclusion is narrower: under the evaluated checkpoint, cohort, and execution protocol, continuation behavior remains sensitive to the history-formed prefix cache after the boundary fast weights are matched. Additional closure diagnostics and cross-model prefix-cache replications are reported in Appendix E, with execution-validity checks summarized in Appendix G.

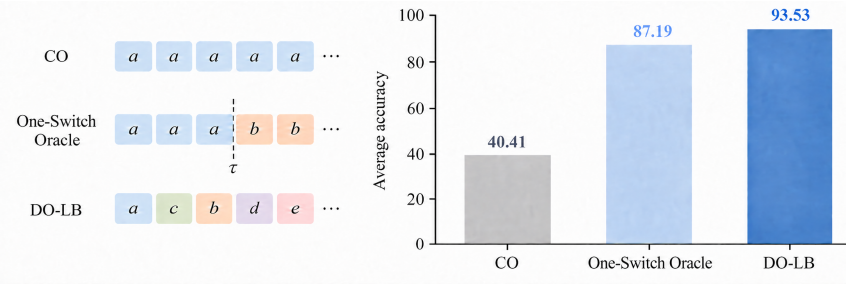


Figure 4: **One-Switch Oracle alternatives.** Left: CO, One-Switch Oracle, and Dynamic Oracle trajectories. Right: Average accuracy on the 170 improved examples.

5 A Large Dynamic Space but a Simple Effective Structure

The sequential comparison establishes that some improved trajectories cannot be represented by any constant action, while the preceding interventions show that their continuation depends on carried history. These findings leave open how much switching complexity is required to reproduce the observed task score. We therefore examine whether trajectories with exactly one action switch can match the scores found in the larger dynamic search space. We call this restricted family the One-Switch Oracle (OSO).

5.1 Experimental Setup

For an improved example x with T_x complete chunks, let $\mathcal{A}$ denote the seven-action space defined earlier. We enumerate

$$\Pi_{\mathrm{OSO}}(x) = \left\{ \left(A^{\tau}, B^{T_x - \tau}\right) \;\middle|\; A, B \in \mathcal{A},\ A \neq B,\ 1 \leq \tau < T_x \right\}. \tag{8}$$

Each candidate consists of two nonempty constant-action segments. State accumulated during the first segment is carried across the switch, without resetting the fast weights. The best score in this family is

$$R_{\mathrm{OSO}}(x) = \max_{\pi \in \Pi_{\mathrm{OSO}}(x)} R(x, \pi). \tag{9}$$

Each example contributes $42(T_x - 1)$ candidates. Exhaustive enumeration evaluates $97{,}356$ candidates over the $N = 170$ dynamic-improved examples, using the same checkpoint, scorer, and execution protocol as the DO trajectories. Because selection uses the final task score, OSO is a restricted oracle analysis rather than an answer-free online policy.

5.2 Results

As shown in Figure 4, the best one-switch trajectories recover much of the score gap between *CO* and the DO dynamic trajectories, leaving a mean Dynamic Oracle–OSO difference of 6.34 percentage points.

The overall $154/170$ match count combines two cases. The 92 DO trajectories that already contain exactly one switch belong to the OSO family by construction, and all 92 are score-matched. This serves as a family-inclusion check. The nontrivial comparison concerns the 78 DO trajectories containing multiple switches: $62/78$ (79.49%) admit an exactly-one-switch alternative with the same final task score. The remaining 16 retain a positive score gap to the one-switch family.

Thus, multiple switches are unnecessary for reproducing the observed score on most of the multi-switch cohort, although one switch does not represent every successful trajectory. Detailed recovery ratios, candidate multiplicity, switch-boundary analyses, and cross-model OSO replications are reported in Appendix F.

These results establish a score-level simplification within the tested action space. They do not establish that a matched OSO trajectory reproduces the Dynamic Oracle generated text or internal state, that one switch is sufficient for every useful trajectory, or that the DO trajectory is globally optimal outside the enumerated OSO family.

6 Related Work

From test-time adaptation to within-example TTT. Test-time adaptation (TTA) addresses changing test conditions through self-supervised training (Sun et al., 2020), entropy minimization (Wang et al., 2021), classifier adjustment (Iwasawa & Matsuo, 2021), augmentation (Zhang et al., 2022), continual adaptation (Wang et al., 2022), and anti-forgetting objectives (Niu et al., 2022). These methods mainly target distribution shifts across test samples or streams. Long-context language models pose a complementary intra-example problem: temporary state must be accumulated and used across chunks of one input. TTT addresses this setting by allowing model state to evolve during inference.

TTT substrates and static schedules. Language-model TTT constructs temporary state through neighbor-based adaptation (Hardt & Sun, 2024), recurrent inner learners (Sun et al., 2024), explicit memory such as Titans (Behrouz et al., 2025), large-chunk learners (Zhang et al., 2025), in-place updates (Feng et al., 2026), and end-to-end long-context training (Tandon et al., 2025). Modular TTT organizes this space around the fast-weight network, objective, and update rule (Tang et al., 2026), while TTT-NTP develops predictive update targets (Ouyang et al., 2026). These works mainly study what state is written and how it is updated; the within-example write trajectory is typically fixed or not itself the primary object of study. We instead fix the checkpoint, substrate, objective, and action set, and study which action is applied to which chunk and in what order. Our checkpoint inherits the in-place substrate (Feng et al., 2026) and a next-position target informed by TTT-NTP.

Toward dynamic adaptation control. Recent methods already make adaptation selective: SCoL chooses update locations (Wang et al., 2026), GDWM allocates writes under a budget (Mei et al., 2026), S-TTT selects evidence spans (Zhu et al., 2026), SEAL learns self-edits and update directives (Zweiger et al., 2025), and EASE-TTT couples evidence selection with query-side adaptation (Yuan et al., 2026). Our distinction is therefore not selectivity itself, but a fixed-checkpoint, fixed-action trajectory analysis: we compare non-constant schedules against the per-example best constant action and diagnose action order and carried state. This is an offline oracle analysis of sequential utility and switching complexity, not an online controller. CALM and Mixture-of-Depths instead vary forward computation rather than temporary parameter writes (Schuster et al., 2022; Raposo et al., 2024).

Fast weights, caches, and interventions. Fast weights retain sequence history (Ba et al., 2016; Schlag et al., 2021). KV-binding TTT has been linked to linear attention (Liu et al., 2026), while qTTT adapts query projections while retaining cached keys and values (Bansal et al., 2025); these differ from the Transformer prefix attention cache studied here. Our cache replacement is a diagnostic intervention related to activation patching (Zhang & Nanda, 2024), not a new memory mechanism or evidence that fast weights and caches are independent memories. We interpret its effect only as a downstream continuation contrast under the specified execution protocol, consistent with cautions against inferring persistent deployment memory from local TTT gains (Song et al., 2026).

7 Conclusion

We show that varying write actions within an example can improve chunkwise in-place test-time training beyond the best constant action. On Qwen3-1.7B, our bounded oracle evaluation yields a conservative full-benchmark gain of 1.39 percentage points over the CO. Our interventions show that the observed utility depends on the organization of write actions and the fast-weight state accumulated across chunks. Continuation also remains sensitive to the prefix key–value cache when boundary fast weights and the execution path are matched, showing that fast weights alone do not capture all relevant computational history. Yet complex switching is often unnecessary: exhaustive OSO evaluation identifies exactly score-matched one-switch alternatives for most improved multi-switch trajectories. These findings motivate designing TTT policies around where and when to write, while accounting for the state carried into subsequent computation. The one-switch results suggest a compact starting point for learning action choices and switch boundaries, while the intervention framework provides a way to examine how fast weights and prefix-cache history affect continuation. Together, these perspectives can guide the development and evaluation of adaptive write policies. Learning such policies without reference answers, and testing whether these findings generalize across checkpoints, tasks, and execution protocols, are important next steps.

REPRODUCIBILITY STATEMENT

The supplementary package contains the frozen per-sample score tables, trajectory manifest, and CPU scripts generating this revision's statistics and figures. Model execution additionally requires the historical checkpoint, benchmark inputs, and environment. Their availability and the limits of the public source snapshot are documented in Appendix G.5.

STATEMENT OF AI ASSISTANCE

Generative AI tools were used to assist with manuscript drafting and editing, literature search and summarization, inspection and editing of analysis code, preparation of statistical summaries and scientific figures from existing experimental records, and feedback on the presentation and interpretation of existing results. Generative AI was not used to generate synthetic data, formulate or prove mathematical claims, or execute model-training or model-inference experiments. All AI-assisted analyses, figures, citations, code changes, and manuscript text were reviewed by the authors against the underlying experimental records and source materials. The authors take full responsibility for the final content of the paper, including all claims, results, and artifacts produced with the assistance of generative AI.

REFERENCES

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1610.06258.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Sanjay Kale, and Samy Jelassi. Let's (not) just put things in context: Test-time training for long-context LLMs, 2025. URL https://arxiv.org/abs/2512.13898.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2501.00663.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1/.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Di He, Wenhao Huang, and Tianle Cai. In-place test-time training, 2026. URL https://arxiv.org/abs/2604.06169.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.18466.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/1415fe9fea0fa1e45dddcff5682239a0-Abstract.html.

Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with KV binding is secretly linear attention, 2026. URL https://arxiv.org/abs/2602.21204.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Yuyao Ge, Baolong Bi, Jiayu Yao, Jun Wan, Ziling Yin, Jiafeng Guo, and Xueqi Cheng. Gated differentiable working memory for long-context language modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31885–31913. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1471. URL https://aclanthology.org/2026.acl-long.1471/.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905. PMLR, 2022. URL https://proceedings.mlr.press/v162/niu22a.html.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g2YraF75Tj.

Xuan Ouyang, Zefan Cai, and Junjie Hu. Test-time training with next-token prediction, 2026. URL https://arxiv.org/abs/2606.21803.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021. URL https://arxiv.org/abs/2102.11174.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2207.07061.

Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. EcoTTA: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11920–11929, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Song_EcoTTA_Memory-Efficient_Continual_Test-Time_Adaptation_via_Self-Distilled_Regularization_CVPR_2023_paper.html.

Xiangchen Song, Zhenhao Chen, Lingjing Kong, Shaoan Xie, Xinshuai Dong, Guangyi Chen, and Kun Zhang. Beyond perplexity: A behavioral evaluation framework for deployment-memory claims in LLM test-time training, 2026. URL https://arxiv.org/abs/2607.00368.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states, 2024. URL https://arxiv.org/abs/2407.04620.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Bohao Tang, Zhen Qin, Yuqi Pan, Zheng Li, Pengfei Liu, and Ya Zhang. Modular TTT: Rethinking test-time training as composable modules, 2026. URL https://arxiv.org/abs/2608.07110.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Wang_Continual_Test-Time_Domain_Adaptation_CVPR_2022_paper.html.

Zekun Wang, Anant Gupta, Zihan Dong, and Christopher J. MacLellan. Self-consolidating language models: Continual knowledge incorporation from context, 2026. URL https://arxiv.org/abs/2605.07076.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Xiaopeng Yuan, Zebin Wang, Suwen Wang, Zongxin Yang, Haohan Wang, and Yushun Dong. EASE-TTT: Evidence-aligned selective test-time training for long-context question answering, 2026. URL https://arxiv.org/abs/2606.06906.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16042.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper/2022/hash/fc28053a08f59fccb48b11f2e31e81c7-Abstract-Conference.html.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right, 2025. URL https://arxiv.org/abs/2505.23884.

Xinyu Zhu, Zhe Xu, Xiaohan Wei, Yunchen Pu, Fei Tian, Chonglin Sun, Kaushik Rangadurai, Hua Zhi, Frank Shyu, Sandeep Pandey, Luke Simon, Yu Meng, and Xi Liu. Self-guided test-time training for long-context LLMs, 2026. URL https://arxiv.org/abs/2607.09415.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models, 2025. URL https://arxiv.org/abs/2506.10943.

A EXPERIMENTAL DETAILS FOR THE MECHANISM ANALYSIS

The main mechanism analyses in Sections 3–5 use Qwen3-1.7B. Supplementary mechanism replications on Qwen3-4B and Llama-3.1-8B are reported in Appendices C–F. The cross-model scores in Table 1 remain the complete 13-task aggregates under the fixed model-specific settings described below.

A.1 CROSS-MODEL CONFIGURATIONS AND THE BASE REFERENCE

Table 2 summarizes the inference settings needed to interpret the three model columns in Table 1. All three columns use the same 13-task, 6,500-example full evaluation scope defined in the main text, but the larger-model settings use a longer configured context, larger chunks, and one additional writable layer. The Qwen3-4B and Llama-3.1-8B columns use a 32,768-token configured context, as shown in Table 1. The bounded dynamic search keeps beam width four and one forward coordinate sweep across all three model settings.

Setting	**Qwen3-1.7B**	**Qwen3-4B**	**Llama-3.1-8B**
Full evaluation scope	13 tasks / 6,500	13 tasks / 6,500	13 tasks / 6,500
Context length	16,384	32,768	32,768
Chunk size C	1,024	4,096	4,096
Writable layers	0,6,12,18,24	0,6,12,18,24,30	0,6,12,18,24,30
Constant actions	7	8	8
Dynamic search	beam 4, 1 sweep	beam 4, 1 sweep	beam 4, 1 sweep
Precision / attention	BF16 / SDPA	BF16 / SDPA	BF16 / SDPA
Decoding	greedy	greedy	greedy
Tail / generation writes	disabled	disabled	disabled

Table 2: Model-specific inference settings for the cross-model comparison in Table 1. The table reports the fixed evaluation configuration rather than implying a scale-only ablation with identical chunk or layer choices.

For Qwen3-1.7B, the constant-action space is the seven-action set defined in the main text,

$$\mathcal{A} = \{\text{OFF}, L_0, L_6, L_{12}, L_{18}, L_{24}, \text{ALL}\}.$$

For Qwen3-4B and Llama-3.1-8B, the corresponding action space additionally includes L_{30}, giving eight constant actions. Thus, the CONSTANT ORACLE (CO) is the per-example maximum over the constant actions available in the reported model setting: seven actions for Qwen3-1.7B and eight actions for Qwen3-4B and Llama-3.1-8B. The NON-CEIL. and IMP. cohorts remain model-specific, as stated in Appendix C.

BASE in Table 1 is a separate checkpoint reference, not a write action. For each model, BASE evaluates the corresponding original pretrained/base checkpoint with test-time training disabled. In contrast, OFF evaluates the corresponding continued-trained TTT research checkpoint while suppressing new writes during the example. The two rows therefore answer different questions: BASE measures the original model before the continued-trained TTT checkpoint, whereas OFF measures that continued-trained checkpoint without additional test-time writes. The Base runs use the same frozen benchmark rows, references, scorer, and per-sample generation budgets as the associated model column, with the tokenizer of the corresponding model. BASE is not included in the constant-action pool used to construct CO, is not a candidate in the bounded trajectory search, and does not define the non-ceiling or improved cohorts.

A.2 INHERITED TARGET CONSTRUCTION

The evaluator follows the in-place adaptation family of Feng et al. (2026) and uses a next-position contextual target informed by Ouyang et al. (2026). These are inherited components; the contribution of this paper is the trajectory and state analysis built on top of the fixed checkpoint and update rule.

Let d be the hidden dimension and m the MLP intermediate dimension. For writable layer ℓ, chunk t, and within-chunk position j, let $z_{\ell,t,j} \in \mathbb{R}^d$ denote the normalized MLP input and $u_{\ell,t,j} \in \mathbb{R}^m$ the gated MLP feature. The target entering the outer-product update is

$$v_{\ell,t,j} = P_\ell^\top (g_\ell \odot z_{\ell,t,j+1}), \qquad 1 \le j < C,$$

where $g_\ell \in \mathbb{R}^d$, $P_\ell \in \mathbb{R}^{d\times d}$, and therefore $v_{\ell,t,j} \in \mathbb{R}^d$. The writable down-projection weight has shape $W_\ell \in \mathbb{R}^{d\times m}$, so the outer product $v_{\ell,t,j} u_{\ell,t,j}^\top$ has the same shape as the updated weight. The signed channel vector and projection are learned before evaluation and then frozen. Target pairs are formed only within the currently observed complete chunk; no pair crosses a chunk boundary and no generated token is used as a target.

The update and write-mask semantics are those stated in Section 2. The Frobenius norm used for clipping is evaluated in float32. The clipped increment is then added to the effective BF16 weight. Consequently, the norm of the proposed clipped increment and the norm of the actually represented BF16 weight difference need not be numerically identical. This numerical distinction is relevant to execution audits but does not change the write semantics in the main text.

A.3 Sample isolation and state lifetime

Each evaluated trajectory starts from the same frozen checkpoint state. The fast weights accumulated for one trajectory are temporary and are not written back to the checkpoint or shared with another example. Incomplete input tails and generated tokens read the accumulated state but do not create additional writes, matching the protocol in Section 2.

The final task answer is used only after generation by the scorer and by the answer-aware oracle selection rules. It is not used to construct the next-position target or to modify the forward computation of a candidate. Thus, the selections defining DO-LB and the One-Switch Oracle (OSO) are answer-aware oracle analyses rather than deployable answer-free controllers.

The execution-path checks that support the common-path assumption used in the mechanism experiments are reported separately in Appendix G; they are not inferred from the clipping threshold or from the equations above.

B Search and Statistical Protocol

B.1 Bounded dynamic search

The search starts from the seven constant trajectories defined in Section 2 and performs one forward coordinate sweep. The initial beam therefore contains all seven constant seeds. At the first coordinate, all seven seeds are expanded before pruning; after that step, and after every later coordinate, the active beam is reduced to width four. At each position, every active member is expanded with all seven actions, duplicate complete trajectories are removed, and newly encountered candidates are scored after generation.

Beam pruning first ranks candidates by task score. Within an exact-score group, it retains candidates so as to maximize the minimum Hamming distance to those already selected for that beam, with a fixed deterministic ordering over full action sequences used to break any remaining tie. After the complete sweep, among candidates tied for the best score within the numerical equality tolerance, the reported winner prefers fewer action switches, then fewer distinct actions, followed by the same fixed deterministic sequence ordering.

For the reported evaluator, a newly evaluated candidate is executed by a clean full replay from checkpoint-initialized fast weights and a fresh attention cache. The reported search therefore does not reuse another candidate's cached prefix. The phrase "replay the affected suffix from the corresponding prefix state" in the main text describes the trajectory semantics: after a coordinate change, only the continuation after that prefix can differ conceptually. In the reported evaluation, however, that candidate-specific prefix state is reconstructed by replaying the candidate from the checkpoint. No saved prefix state is copied or reused across candidate trajectories.

The result pool retains the constant trajectories even when they leave the active beam, ensuring that the best retained score cannot fall below the CONSTANT ORACLE (CO). The selected winner is replayed once more and its score and decoded prediction are checked against the stored record.

B.2 CANDIDATE ACCOUNTING

For the seven-action Qwen3-1.7B search, the first coordinate can add at most six new alternatives for each constant trajectory. At every later coordinate, at most four active beam members can each add six new alternatives. Hence the per-example upper bound stated in the main text is

$$Q_{\max}(T) = 7 + 7(7-1) + (T-1)4(7-1) = 24T + 25. \tag{10}$$

Deduplication can reduce the realized number of unique candidates, and the forced winner replay is a separate execution. This bound is therefore a candidate-accounting bound rather than a wall-clock or model-call estimate.

B.3 EQUAL-BUDGET RANDOM-SEARCH ABLATION

To separate the effect of the structured search rule from the effect of simply scoring many nonconstant trajectories, we compare the bounded coordinate search with a pre-specified uniform random-search realization on the same $N = 170$ Qwen3-1.7B improved examples. Random trajectories are sampled uniformly without replacement from the nonconstant trajectory space. The seven constant trajectories are excluded from the random draw, no early stopping is used, and the checkpoint, action space, execution protocol, decoding, and task scorer are held fixed. The total number of fresh, unique nonconstant trajectory evaluations is matched exactly between the two searches at 62,772.

Search reference	Mean score	Gain over CO (pp)
CO	40.41	0.00
Equal-budget random	89.96	+49.55
DO-LB	**93.53**	**+53.12**

Table 3: Equal-budget search control on the 170 Qwen3-1.7B improved examples. Both nonconstant searches use the same aggregate budget of 62,772 fresh, unique trajectory evaluations.

Uniform random search also locates high-scoring dynamic trajectories, recovering 93.28% of the observed DO-LB gain over CO. This reinforces that the sequential-utility phenomenon is not tied to a single selected path. Under the exactly matched evaluation budget, however, DO-LB remains $3.57\,\mathrm{pp}$ higher on average (paired task-stratified 95% CI $[1.45, 6.04]$). It matches the random result on 159/170 examples and is higher on the remaining 11; the random search is never higher. Thus, the observed advantage is not explained by evaluation count alone, while the structured coordinate search provides an additional score benefit at fixed budget. The interval uses 20,000 paired task-stratified bootstrap draws and reflects sample uncertainty conditional on this fixed random realization.

B.4 PAIRED ESTIMANDS AND BOOTSTRAP INTERVALS

All score differences are computed per example before aggregation. For the main DO-LB–CO contrasts, Reverse, Reset, Selective Deletion, the magnitude-matched nonzero perturbation control, and the prefix-cache intervention, the example is the statistical unit. When an intervention has multiple internal branches, the prespecified branch scores are first combined into one example-level score; in particular, Selective Deletion is averaged within example before the paired difference is formed.

The reported primary confidence intervals use task-stratified paired bootstrap resampling with 20,000 draws and seed 20260921. Resampling preserves the observed number of examples in each task stratum, and the 2.5th and 97.5th percentiles define the reported interval. The intervals are conditional on the frozen checkpoint, task composition, scorer, search output, and cohort definitions; they do not represent retraining or repeated-search variability.

The final task score is also used in the selections defining DO-LB and OSO. Accordingly, these quantities establish oracle-accessible scores within the evaluated candidate families, not the performance of a deployable answer-free controller or the efficiency of the search procedure.

C COHORT DEFINITIONS

The model-scale aggregates are reported in Table 1. The model-specific inference settings and the meaning of the BASE row are given in Appendix A.1. This appendix only records the Qwen3-1.7B cohort lineage used by the mechanism analyses; it does not repeat the main score table.

Cohort	N	Definition and use
All examples	6,500	Complete 13-task evaluation scope.
Non-ceiling	1,933	$R_{\mathrm{CO}} < 1$; bounded-search scope in the main text.
Improved	170	DO-LB>CO; Reverse and OSO.
Structurally eligible	165	Frozen trajectory-only rule; Reset and prefix-cache intervention.
Nontrivial prefix	157	Structurally eligible minus eight all-OFF prefixes; Selective Deletion and matched control.

Table 4: Qwen3-1.7B cohort lineage used by Sections 3–5.

The 165-example structural screen is applied only after the 170 improved trajectories have been fixed and does not inspect any intervention outcome. The exact structural rule and the reason for the 165-to-157 reduction are given in Appendix D.

The NON-CEIL. and IMP. columns in Table 1 are defined separately for each model. The BASE scores are checkpoint references only: they do not enter the definitions of CO, non-ceiling, improved, or any mechanism cohort in this appendix.

C.1 CROSS-MODEL MECHANISM COHORTS

We additionally replicate the mechanism analyses on Qwen3-4B and Llama-3.1-8B under their 32K-token, 4K-chunk settings. The corresponding improved cohorts contain the examples used by each cross-model mechanism replication for which DO-LB>CO.

Model	Improved	Reset / KV	Selective Deletion
Qwen3-4B	85	69	62
Llama-3.1-8B	39	30	29

Table 5: Conditional cohorts used by the cross-model mechanism replications. The Improved cohort is defined by DO-LB>CO.

Because the 32K configurations use 4K-token chunks and therefore contain far fewer complete chunks per example, the cross-model replications use a resolution-aware version of the structural screen. An improved trajectory is eligible when its first switch position satisfies

$$s_x \leq \max(2, \lfloor T_x/4 \rfloor)$$

and the modal action occupies at least 80% of the second-half positions. The intervention boundary remains the start of the final contiguous run of the selected DO-LB final action. This screen depends only on the selected trajectory and is fixed before any intervention outcome is inspected.

D Structural Eligibility and Intervention Details

D.1 Frozen structural rule and boundary

For an improved Qwen3-1.7B trajectory $\pi_x^{\mathrm{N}} = (a_1^{\mathrm{N}}, \ldots, a_{T_x}^{\mathrm{N}})$, define the first switch

$$s_x = \min\{i \in \{2, \ldots, T_x\} : a_i^{\mathrm{N}} \neq a_{i-1}^{\mathrm{N}}\}.$$

Let the second-half positions be $H_x = \{\lfloor T_x/2 \rfloor + 1, \ldots, T_x\}$. The trajectory is structurally eligible when

$$\frac{s_x}{T_x} \leq 0.25, \qquad \max_{a \in \mathcal{A}} \frac{1}{|H_x|} \sum_{i \in H_x} \mathbf{1}[a_i^{\mathrm{N}} = a] \geq 0.80.$$

This rule depends only on the selected DO-LB action trajectory and is frozen before Reset, prefix-cache, or Deletion outcomes are read. Applied to the 170 improved trajectories, it yields the 165-example cohort used by Reset and the prefix-cache intervention.

The intervention boundary is the start of the final contiguous run of the selected DO-LB final action,

$$b_x = 1 + \max\{i < T_x : a_i^{\mathrm{N}} \neq a_{T_x}^{\mathrm{N}}\}.$$

It is immediately before chunk b_x and need not coincide with the first switch or the midpoint of the trajectory.

D.2 Reset and the nontrivial-prefix subset

Reset restores every writable fast weight at b_x^- to its checkpoint value while retaining the native prefix cache and native suffix actions. The suffix is then recomputed under the reset state, as described in Section 3.

Eight of the 165 structurally eligible examples have an all-OFF prefix before b_x. These examples accumulate no pre-boundary fast-weight write, so their native boundary fast weights already equal the checkpoint weights and Reset is trivial. Excluding them yields the 157-example nontrivial-prefix cohort used by Selective Deletion:

$$165 - 8 = 157.$$

The exclusion is determined from the native prefix actions and does not use Reset or Deletion outcomes.

D.3 Selective Deletion branch rule

Selective Deletion removes the accumulated boundary offset of a prescribed writable layer rather than deleting a single earlier write event. If the native prefix contains no ALL action, the first non-OFF single-layer action determines the target layer. If the prefix contains ALL, each writable layer is evaluated in a separate deletion branch. For a branch targeting layer ℓ, only that layer is restored to its checkpoint value at the boundary; the other boundary weights, native prefix cache, and native suffix actions are retained.

For an example with target-layer set J_x, the reported Deletion score is the within-example average

$$\overline{R}_{\mathrm{Del}}(x) = \frac{1}{|J_x|} \sum_{\ell \in J_x} R_{\mathrm{Del},\ell}(x).$$

The paired native–Deletion contrast is formed only after this averaging, so examples with an ALL prefix do not receive extra statistical weight.

D.4 Magnitude-matched nonzero perturbation control

To test whether the Deletion drop can be explained by perturbation magnitude alone, we evaluate an approximately magnitude-matched nonzero control on the same 157-example nontrivial-prefix cohort and the same 349 target-layer branches defined above. For each branch, the deleted boundary offset is replaced by a random perturbation constructed without access to the final task score, whose realized weight-change norm is matched as closely as possible to the norm of that deleted offset. Five

frozen random perturbations are evaluated per branch, yielding 1,745 branch-level perturbations in total. Of these, 1,744/1,745 are matched within 1% relative norm error; the single remaining case has 1.1731% relative error. No sample, branch, or perturbation is selected using task score.

For each example, we first average the five random perturbations within each target-layer branch and then apply the same within-example branch averaging as for Selective Deletion. The resulting mean scores and paired contrasts are:

Statistic	Estimate	95% CI
DO-LB score	93.25	–
Selective Deletion score	54.31	–
Magnitude-matched random score	58.85	–
DO-LB − matched random	34.40 pp	[31.50, 37.25]
Matched random − Deletion	4.54 pp	[1.67, 7.35]

Table 6: Magnitude-matched nonzero perturbation control on the 157-example Selective Deletion cohort. Scores are on the 0–100 task-score scale; confidence intervals are paired at the example level.

The approximately magnitude-matched random perturbation remains far below the selected DO-LB trajectory, showing that a generic nonzero perturbation of this scale is itself disruptive. At the same time, it scores 4.54 percentage points above Selective Deletion, with a paired 95% confidence interval that excludes zero. Thus, perturbation magnitude alone does not explain the full Deletion effect under this control. This comparison still does not establish that the deleted trajectory-formed direction is uniquely causal or that arbitrary directions of the same norm would produce the same contrast.

D.5 Cross-model intervention replication

We repeat Reverse, Reset, and Selective Deletion on the Qwen3-4B and Llama-3.1-8B cohorts in Appendix C.1. Reverse uses the complete selected DO-LB action sequence. Reset restores all writable fast weights at the prescribed boundary, while Selective Deletion restores only the prescribed layer and retains the native prefix cache and suffix actions. For an ALL prefix, Selective Deletion evaluates the six writable layers separately and averages them within the example.

Model	Intervention	N	DO-LB	Intervention	Drop (pp)
Llama-3.1-8B	Reverse	39	100.00	44.87	55.13
	Reset	30	100.00	47.39	52.61
	Selective Deletion	29	100.00	49.26	50.74
Qwen3-4B	Reverse	85	97.02	43.02	54.00
	Reset	69	96.33	47.25	49.08
	Selective Deletion	62	96.45	43.91	52.54

Table 7: Cross-model replication of the schedule and fast-weight interventions. Scores are conditional mean task scores on the 0–100 scale.

The intervention directions are consistent across the two additional models. Reverse decreases the score on 36/39 Llama examples and 76/85 Qwen examples; Reset decreases 26/30 and 58/69; Selective Deletion decreases 25/29 and 59/62. No example improves under these interventions.

Reverse, Reset, and Selective Deletion are different interventions on different conditional cohorts. Their drop magnitudes are therefore not additive and should not be read as directly comparable causal effect sizes. The magnitude-matched control is used only to calibrate the interpretation of Selective Deletion on the same 157-example cohort.

E PREFIX-CACHE EXECUTION DIAGNOSTICS

Section 4 defines the 165-example prefix-cache intervention and its score contrast. This appendix records only the additional execution checks used to interpret that contrast; it does not repeat the main result table.

E.1 DONOR AND BOUNDARY CHECKS

The Dynamic Oracle donor and the all-OFF-prefix control donor provide only cache positions strictly before the prespecified boundary b_x. Both keys and values at all attention layers are included in the donor cache. At injection, the two branches receive the same Dynamic Oracle boundary fast weights and the same Dynamic Oracle suffix actions. No post-boundary Dynamic Oracle cache entry or Dynamic Oracle fast-weight update is copied into the control branch; the continuation is recomputed independently after injection.

Thus, equality of the fast weights is a boundary condition only. Different prefix caches may induce different suffix activations and therefore different later fast-weight updates. The 14.56 pp contrast in the main text is the total downstream effect of changing the starting prefix cache under this boundary condition, not a direct cache effect with the later computation clamped.

E.2 RECORDED CLOSURE DIAGNOSTICS

All closure diagnostics compare the corresponding intervention replay against an independently replayed Dynamic Oracle trajectory for the same sample. The "exact-trace" criterion requires bitwise equality for every tensor in the declared trace set. For Qwen3-1.7B, this set records, at all 28 attention layers, the attention input, query/key/value projections, attention output, MLP input and output, and residual for the first complete suffix chunk; the remainder is recorded when present, together with the final continuation norm and the prompt logits returned by the prefill pass. Decoded-prediction agreement is evaluated separately from this tensor-level trace criterion.

On the 165-example cohort, the control-history $\mathcal{K}_0$ branch matches the Dynamic Oracle replay on the complete declared trace set for 110/165 examples and on the decoded prediction for 118/165. The Dynamic-Oracle-history $\mathcal{K}_1$ replay matches the Dynamic Oracle replay on the declared trace set for 165/165 examples and also matches the Dynamic Oracle decoded prediction and task score on 165/165. These diagnostics do not establish equality of every raw or clipped post-boundary update, every post-update fast-weight tensor, or every generation-step activation; no such broader closure claim is made here.

E.3 CROSS-MODEL PREFIX-CACHE REPLICATION

The matched-boundary prefix-cache intervention is repeated on the eligible Qwen3-4B and Llama-3.1-8B cohorts in Appendix C.1. In both branches, the Dynamic Oracle boundary fast weights and suffix actions are matched; only the pre-boundary prefix cache is taken from the control or Dynamic Oracle history.

Model	N	Control KV ($\mathcal{K}_0$)	DO KV ($\mathcal{K}_1$)	$\mathcal{K}_1 - \mathcal{K}_0$ (pp)
Qwen3-4B	69	88.55	96.33	+7.78
Llama-3.1-8B	30	99.33	100.00	+0.67

Table 8: Cross-model replication of the matched-boundary prefix-cache intervention. Scores are conditional means on the 0–100 task-score scale.

The Dynamic Oracle history cache is non-decreasing on every evaluated example in both models. Qwen3-4B improves on 10/69 examples and ties on 59/69, whereas Llama-3.1-8B improves on 1/30 and ties on 29/30. Together with the 14.56 pp Qwen3-1.7B contrast in the main text, the replication shows the same direction of prefix-history sensitivity while its magnitude varies across model settings.

The common-path and future-action invariance checks that address whether the trajectory itself can alter the numerical route before the action differs are reported in Appendix G.

F One-Switch Oracle Supplementary Analyses

The main text defines the One-Switch Oracle (OSO) family and reports the primary result: 92 selected DO-LB trajectories are already one-switch, while 62 of the 78 selected multi-switch trajectories admit an exactly-one-switch alternative with the same final task score. This appendix reports the supplementary recovery, multiplicity, and switch-order diagnostics referenced in Section 5.2.

F.1 Gain recovery

For an improved example, define the score-recovery ratio

$$\rho_x = \frac{R_{\mathrm{OSO}}(x) - R_{\mathrm{CO}}(x)}{R(x, \pi_x^{\mathrm{N}}) - R_{\mathrm{CO}}(x)}.$$

The denominator is the gain of the selected DO-LB trajectory over CO. The ratio is reported without clipping. In the observed 170-example cohort it takes value 1 for the 154 examples whose best OSO score matches the DO-LB score and value 0 for the remaining 16, whose best OSO score equals CO. The mean recovery is therefore 90.59%, with a task-stratified 95% CI of [85.88%, 94.71%]. The aggregate recovery ratio obtained after summing gains before division is 88.06%. These are conditional score-recovery summaries, not population success rates.

The supplementary OSO intervals in this appendix use 4,000 task-stratified bootstrap draws with seed 20260923. They are separate from the 20,000-draw primary intervals described in Appendix B.4.

F.2 Multiplicity and switch-position sensitivity

Among the 62 selected multi-switch examples with a score-matched OSO alternative, 50 have multiple score-matched one-switch candidates and 12 have exactly one; the median number of matching candidates, computed over these 62 examples, is four. For 22/62 examples, at least one ordered action pair has two adjacent switch boundaries that both match the DO-LB score. In the remaining 40/62, every successful switch boundary is isolated from other successful boundaries within the same ordered action pair. Thus, low score-level switching complexity does not imply that the switch boundary is arbitrary.

F.3 Reversal sensitivity of one-switch solutions

The reversal diagnostic compares $(A^\tau, B^{T-\tau})$ with its count-matched reversal $(B^{T-\tau}, A^\tau)$. Within each example, "disagreement" means that the two task scores differ by more than 10^{-12}; it does not refer to decoded-prediction disagreement. The reported 9.92% disagreement rate and 4.66 pp mean absolute score difference are computed by first summarizing reversal pairs within each example and then averaging the 170 example-level summaries equally; they are not pooled over all reversal pairs. The same 10^{-12} task-score tolerance is used when calling an OSO candidate score-matched to the DO-LB score.

Within the 62 matched multi-switch examples, every example has at least one score-matched one-switch candidate that scores strictly above its reversal. Of these 62 examples, 54 have the stronger property that *all* of their score-matched candidates score strictly above their respective reversals. Across the 535 score-matched candidates from those 62 examples, 513 are strictly above their reversals. These conditional comparisons support the main text's statement that a one-switch score match does not make timing and order irrelevant; they are not an unselected causal estimate of ordering effects.

F.4 CROSS-MODEL OSO REPLICATION

We exhaustively evaluate the corresponding One-Switch Oracle family on the Qwen3-4B and Llama-3.1-8B improved cohorts. Because these configurations use eight actions, each internal switch boundary admits $8 \times 7 = 56$ ordered action pairs.

Model	N	CO	DO-LB	OSO	Exact match
Qwen3-4B	85	37.14	97.02	93.51	79/85
Llama-3.1-8B	39	40.77	100.00	96.58	37/39

Table 9: Cross-model One-Switch Oracle results on the improved cohorts. DO-LB denotes the selected bounded-search trajectory. Scores are conditional means on the 0–100 task-score scale.

The best OSO trajectory exactly matches the DO-LB score on 79/85 Qwen3-4B examples and 37/39 Llama-3.1-8B examples. Relative to CO, the OSO family retains 94.1% and 94.2% of the aggregate DO-LB gain, respectively. Exhaustive enumeration evaluates 28,560 Qwen3-4B and 12,992 Llama-3.1-8B one-switch candidates. These results extend the score-level simplification observed for Qwen3-1.7B to the two larger model settings.

G EXECUTION VALIDITY AND SCOPE

G.1 UNIFIED FUNCTIONAL EXECUTION PATH

The reported mechanism experiments keep all five Qwen3-1.7B candidate writable layers on the same functional execution path for every trajectory. The action changes only the chunkwise write mask described in Section 2; it does not select a different numerical route according to the complete trajectory or to future actions. This prevents future write choices from changing prefix computation before those choices occur.

The execution order is read-then-update: a chunk is evaluated using the fast-weight state available at its entrance, and an allowed write is applied only for later computation. Each trajectory candidate used by the bounded search is evaluated from the common checkpoint state with a fresh attention cache. Throughout Appendices B and G, a "candidate prefix state" therefore means the state reached by replaying that candidate up to the corresponding point; it never denotes a saved prefix copied from another candidate. The reported search does not depend on an assumed equivalence between full replay and prefix-state reuse. The selected DO-LB trajectory is replayed once more to verify its stored score and decoded prediction.

G.2 EXECUTION-VALIDITY CHECKS

The execution checks directly test the assumptions most relevant to the schedule and prefix-cache experiments. The recorded results are:

Check	Result	What it establishes
Future-action prefix invariance	15/15 exact	Changing only later actions leaves the recorded common-prefix tensors exactly unchanged.
Zero-write action-path parity	64/64 exact	Across 32 samples, forcing zero writes removes action-dependent numerical differences on the tested path.
Repeated same-path execution	4/4 exact	Repeated executions agree exactly on the recorded comparison.
Read-then-update timing	5/5 layers pass	The five candidate writable layers satisfy the apply-after-forward semantics used in the main text.

Table 10: Execution checks for the unified functional path. "Exact" refers to equality on the quantities recorded for the corresponding check; it is not a claim of universal equality for every unrecorded internal tensor.

Together, these checks support the common-path and read-then-update assumptions used by the reported mechanism analyses. They directly test the execution properties required by the schedule and prefix-cache comparisons in Sections 3 and 4.

These checks do not establish a general equivalence between arbitrary alternative model implementations, nor is such an equivalence required for the reported comparisons because all candidate trajectories use the same functional path. Likewise, no equivalence between full replay and prefix reuse is assumed by the bounded search.

G.3 Numerical precision boundary

The update norm is computed before the BF16 state addition, so a proposed clipped increment and the actually represented BF16 weight difference are not interchangeable quantities. For this reason, execution validity is supported by direct parity checks rather than inferred from the small clipping threshold. The paper does not claim implementation-independent numerical equality across arbitrary kernels, dtypes, or hardware.

G.4 Cross-model protocol scope

The model-specific settings summarized in Table 2 are fixed separately for each model rather than inferred from model size. Qwen3-4B and Llama-3.1-8B use their own fixed-action, bounded-search, and original-BASE evaluations. These settings are kept distinct because the larger-model protocols change context length, chunk size, writable layers, and action count rather than changing model scale alone.

The original-BASE evaluations disable test-time training. Accordingly, the cross-model results test the same trajectory-level questions under the stated model-specific protocols rather than constituting a scale-only ablation.

The supplementary mechanism replications for Qwen3-4B and Llama-3.1-8B use their 32K-token, 4K-chunk model-specific protocols. Their Reverse, Reset, Selective Deletion, prefix-cache, and OSO results are conditional replications under those protocols rather than pooled estimates with the Qwen3-1.7B mechanism cohort.

G.5 Scope of reproducibility

The supplementary materials contain the per-example scores, selected trajectories, per-sample intervention records, and CPU analysis materials needed to regenerate the reported aggregate scores, confidence intervals, cohort counts, OSO summaries, and manuscript figures from the reported out-

comes. This supports statistical reproduction of the reported results without requiring a new model-inference run.

Full model re-execution additionally requires the corresponding continued-trained checkpoints, benchmark inputs, tokenizer assets, and generation environment to be matched to the model-specific settings in Appendix A.1. We therefore distinguish reproduction of the reported statistics from end-to-end re-execution of the model experiments.

The checks in Table 10 support the specific common-path and timing assumptions used by Sections 3 and 4. They do not establish a universal mechanism independent of the evaluated checkpoint and execution protocol, nor do they convert the answer-aware selections defining DO-LB and OSO into deployable online controllers.