LEARNING WHETHER TO LEARN: PROSPECTIVE COUNTERFACTUAL TEST-TIME LEARNING FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time learning (TTL) promises models that improve during deployment, yet the dominant design pattern is myopic: an experience produces an update, and the update is judged by whether it fits that same experience. This creates a structural failure mode under unlabeled or weakly supervised streams—updates can reduce the current loss while degrading future decisions, amplifying pseudo-label noise, interference, and prompt-injection effects. We propose **FORESIGHT-TTL**, a prospective formulation in which a candidate update is valued by its *counterfactual effect on future deployment utility*. During meta-training, matched rollout pairs estimate the counterfactual utility difference between applying and skipping the same update. A compact value model learns this future utility from the current experience, update geometry, uncertainty, and interference features. At deployment, a lower-confidence-bound gate decides *whether* to learn, a plasticity router decides *where* to write, and a reversible fast-weight transaction decides *whether the write should survive*. We further introduce multi-timescale persistence, promoting only repeatedly useful updates from episodic to persistent state. We analyze how calibrated value intervals can bound the probability of accepting harmful updates under a simultaneous calibration assumption. We evaluate the approach primarily on DeepSeek-R1-Distill-Qwen-7B across AIME 2024/2025, MATH-500, GPQA-Diamond, BIG-Bench Hard, and ARC-AGI-2, and additionally test cross-backbone deployment behavior on Qwen3-8B and Llama-3.1-8B-Instruct, with non-IID streams, corruption stress tests, and retention probes.

1 INTRODUCTION

Large language models are increasingly asked to operate in non-stationary environments: a user reveals preferences over many interactions, a solver encounters a novel family of problems, or an agent discovers a new tool convention after deployment. Test-time learning (TTL) addresses this setting by allowing some adaptive state to change during inference rather than freezing the model after pretraining. Recent results make the opportunity concrete. Parameter updates at test time can dramatically improve few-shot abstract reasoning (Akyürek et al., 2025); Zuo et al. (2025) show that unlabeled test questions can drive reinforcement learning through self-consistency; and new fast-weight mechanisms make in-place adaptation practical for released language models (Feng et al., 2026; Zhang et al., 2026; Ouyang et al., 2026). TTL is therefore moving from a conceptual possibility toward a deployment primitive.

The same progress exposes a more fundamental problem: *what makes a test-time update trustworthy?* In supervised training, an update can be assessed against a held-out distribution and undone before deployment. At test time, the update is produced inside the deployment stream, often from the model's own pseudo-label, reward, or reconstruction objective. A locally plausible update can therefore be globally harmful. Test-time RL is particularly vulnerable: noisy pseudo-labels create an "ambiguity region," and group-relative optimization can amplify spurious reward signals (Yu et al., 2026). Separate safety analyses show that adversarial test experiences can amplify harmful behavior while degrading reasoning ability (Khattar et al., 2026). Continual TTL also introduces interference: an update useful for the current domain may damage capabilities needed hundreds of interactions later.

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.

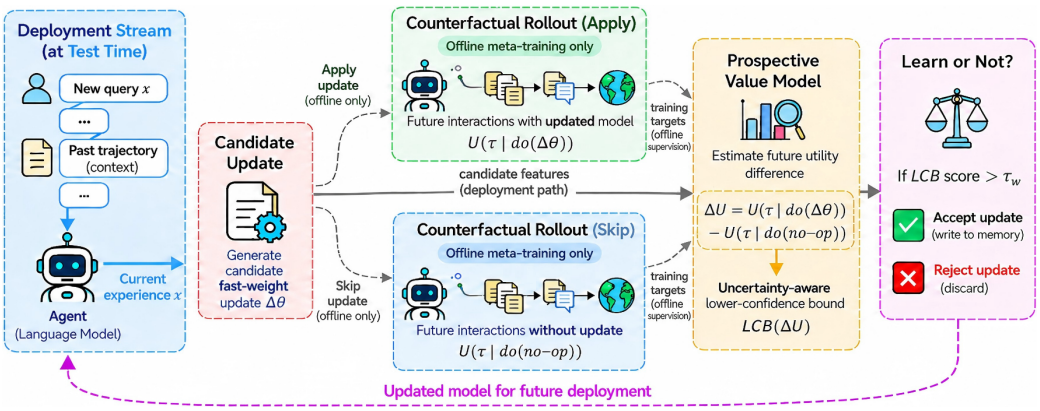

Figure 1: **FORESIGHT-TTL.** Candidate fast-weight updates are scored by a learned prospective utility model trained from matched counterfactual rollout pairs. At test time, a confidence-aware gate chooses whether to learn, a router chooses where to write, and a reversible transaction controls persistence.

Existing methods mostly attack this problem indirectly. Confidence filtering and denoising improve the quality of immediate pseudo-labels (Yu et al., 2026); curricula synthesize learnable variants of hard test questions (Yang et al., 2026; He et al., 2026); parameter-efficient TTT decides how and where to store context (Chen et al., 2026; Feng et al., 2026); and explicit memories can be optimized for long-horizon interaction reward (Xu et al., 2026; Cai et al., 2026). These are valuable advances, but they leave the core causal question unresolved: *if we intervene on the model by applying this specific parameter update now, what happens to future decisions compared with not applying it?*

We propose **FORESIGHT-TTL**, a prospective approach to TTL that treats every candidate update as a decision whose value is defined by its downstream consequences. Let Δ_t be an update proposed after experience e_t. We define its *counterfactual update value* as the discounted difference in future reward between two matched worlds, one in which Δ_t is applied and one in which it is skipped. During meta-training, paired rollout branches provide supervision for a compact value predictor Q_ψ. During deployment, Q_ψ evaluates candidate updates before they are allowed to alter adaptive state. A lower-confidence-bound gate decides whether to learn; a plasticity router chooses the parameter subspace; and a reversible transaction keeps the write provisional until subsequent evidence supports committing it. Useful updates can be promoted from short-lived episodic weights to persistent test-time memory, while harmful or obsolete updates expire or roll back.

Figure 1 summarizes the resulting shift. Conventional TTL implements $e_t \to \Delta_t \to \theta_{t+1}$. FORESIGHT-TTL inserts a prospective decision layer between proposal and write: the model asks not only whether an experience can reduce a local loss, but whether the corresponding update deserves to survive after the triggering example disappears.

Our contributions are fourfold. **First**, we formulate TTL as a deployment-time decision problem and define the value of an update through its counterfactual effect on future utility rather than current fit. **Second**, we introduce a practical paired-rollout estimator and a learned update-value model that predicts this future effect from experience, gradient, uncertainty, and interference features. **Third**, we develop selective plasticity routing, reversible commits, and multi-timescale persistence, producing a single mechanism for deciding *whether, where, and how long* to learn. **Fourth**, we propose evaluation metrics and stress tests that separate adaptation from harmful-update rate, retention, corruption robustness, and value calibration. The resulting thesis is simple: *deciding whether a proposed update should survive is an important challenge in test-time learning.*

2 RELATED WORK

Test-time training and reasoning. Test-time training adapts parameters using signals derived from the test input (Sun et al., 2020). In LLMs, temporary parameter updates improve abstract few-shot reasoning (Akyürek et al., 2025); PERK meta-learns LoRA context memories (Chen et al.,

2026); In-Place TTT and TTT Done Right make fast weights compatible and efficient for transformer LMs (Feng et al., 2026; Zhang et al., 2026); and TTT-NTP aligns writes with next-token prediction (Ouyang et al., 2026). For reasoning, TTRL learns from unlabeled test questions via self-consistency reward (Zuo et al., 2025), DDRL suppresses spurious pseudo-label signals (Yu et al., 2026), and TTCS/TTSR synthesize curricula or targeted variants (Yang et al., 2026; He et al., 2026). ORCA adapts a calibration module for reasoning-time sampling (Zhou et al., 2026). These methods primarily improve *how an update is produced*; FORESIGHT-TTL asks whether a proposed update has positive downstream value before it is allowed to persist.

Continual memory and robust adaptation. EvoLib accumulates reusable abstractions with long-term value weighting (Xu et al., 2026), while MemoPilot learns a memory-update policy for sequential interaction (Cai et al., 2026). Our object of credit is different: a specific parameter intervention is compared against skipping that intervention under a matched future stream, then routed or rolled back. This connects to continual test-time adaptation work on drift and forgetting (Wang et al., 2021; 2022; Niu et al., 2022) and to confidence sequences for sequential uncertainty control (Howard et al., 2021). Recent safety analysis further shows why write control matters: test-time RL can amplify harmful behavior and incur a reasoning tax under malicious inputs (Khattar et al., 2026).

3 PROSPECTIVE TEST-TIME LEARNING

We model deployment as a stream of experiences $e_t = (x_t, \tau_t, o_t)$ arriving from a potentially non-stationary process $\mathcal{D}_t$. Here x_t is the current input, τ_t denotes the model's generated trajectory, and o_t contains any self-supervised or environment feedback available at test time. The model has frozen slow parameters θ and adaptive state w_t containing parameter-efficient adaptive components such as LoRA states. A TTL rule proposes an update Δ_t and transitions

$$w_{t+1} = \mathcal{U}(w_t, \Delta_t, e_t). \tag{1}$$

The deployed predictor is π_{θ, w_t}.

Conventional TTL constructs Δ_t to improve a local objective ℓ_t, e.g., reconstruction, next-token prediction, entropy, or a pseudo-label reward:

$$\Delta_t = -\eta P \nabla_w \ell(e_t; \theta, w_t). \tag{2}$$

The local decrease $\ell_t(w_t) - \ell_t(w_t + \Delta_t)$ is observable, but it is not the deployment objective. We instead define a horizon-H future return

$$G_{t,H}(w) = \sum_{k=1}^{H} \gamma^{k-1} R(e_{t+k}; \theta, w), \tag{3}$$

where R may be answer correctness, task reward, or a calibrated proxy unavailable to the learner but available for meta-training/evaluation. The **counterfactual update value** is

$$\boxed{V_t(\Delta_t) = \mathbb{E}[G_{t,H}(w_t + \Delta_t) - G_{t,H}(w_t) \mid \mathcal{H}_t]\,,} \tag{4}$$

with history $\mathcal{H}_t = (e_{1:t}, w_t)$. Equation 4 measures the counterfactual utility difference of the update, under a matched future stream construction.

The distinction produces four qualitatively different cases. A local improvement with $V_t > 0$ is genuinely useful learning. A local improvement with $V_t < 0$ is *harmful overfit*: the update fits its source experience but damages future utility. Conversely, an update may initially worsen the local loss yet create useful features that yield $V_t > 0$. TTL systems that gate only by current confidence or loss cannot distinguish these cases.

We therefore cast deployment-time adaptation as a meta-MDP. The state is $s_t = (\mathcal{H}_t, w_t)$ and the learning action is

$$a_t = (z_t, j_t, \Delta_t), \tag{5}$$

where $z_t \in \{0, 1\}$ decides whether to write and j_t selects a plastic subspace. Persistence is governed by the reversible state-transition mechanism described in Section 4. The learning policy maximizes

$$\max_{\pi} \; \mathbb{E}_{\pi} \sum_{t} \gamma^t \Big[R_t - \lambda_c C(a_t) - \lambda_i I(a_t) \Big], \tag{6}$$

where C is a normalized candidate write-cost proxy and I penalizes interference with protected capabilities. This formulation separates *task action* (solving the current input) from *learning action* (changing the solver for future inputs).

4 FORESIGHT-TTL

FORESIGHT-TTL instantiates Equation 6 with five components: candidate updates, paired counterfactual supervision, a prospective value model, selective plasticity routing, and reversible multi-timescale persistence.

4.1 CANDIDATE UPDATES FROM WEAK TEST-TIME SUPERVISION

For an experience e_t, we form a weak test-time objective by combining self-consistency, next-token prediction, and trajectory regularization:

$$\ell_t = \alpha \ell_{\mathrm{sc}} + (1-\alpha)\ell_{\mathrm{ntp}} + \lambda_{\mathrm{kl}} D_{\mathrm{KL}}(\pi_{\theta,w_t} \| \pi_{\theta,w_0}). \tag{7}$$

For reasoning questions, ℓ_{sc} treats the majority final answer from K sampled trajectories as a pseudo-label, following the signal used in TTRL (Zuo et al., 2025), while ℓ_{ntp} reuses the causal next-token signal emphasized by TTT-NTP (Ouyang et al., 2026). We never assume that either signal is correct; they only generate candidate updates.

We expose M plastic subspaces $\{S_j\}_{j=1}^{M}$ as low-rank adapters attached to selected attention and MLP projections. All candidates share the same base gradient computation; candidate-specific updates are obtained by inexpensive projections rather than independent backward passes. For subspace j,

$$\Delta_t^{(j)} = -\eta_j P_j \nabla_w \ell_t, \tag{8}$$

where P_j projects the gradient into S_j. We additionally include a no-write candidate $\Delta_t^{(0)} = 0$ and define the normalized candidate cost as $C_j = |S_j| / \max_{k \geq 1} |S_k|$ for $j \geq 1$, with $C_0 = 0$, where $|S_j|$ is the number of adaptive LoRA coefficients touched by subspace S_j. This candidate set lets the method distinguish a poor learning signal from a poor storage location.

4.2 MATCHED COUNTERFACTUAL UPDATE CREDIT

Directly observing $V_t(\Delta)$ at deployment would require replaying the future under both updated and non-updated models. We therefore learn update value offline by constructing paired branches during meta-training. At time t, both branches share the same history, future minibatch $B_{t+1:t+H}$, sampling seeds, and task-level randomness. The update branch receives $\Delta_t^{(j)}$ and the matched baseline branch skips it:

$$w_t^+ = w_t + \Delta_t^{(j)}, \qquad w_t^0 = w_t. \tag{9}$$

$$\hat{V}_t^{(j)}{}_{\mathrm{target}} = G_{t,H}(w_t^+) - G_{t,H}(w_t^0), \tag{10}$$

where $G_{t,H}(w)$ denotes the discounted deployment utility collected over the next H interactions under adaptive state w. Pairing reduces variance caused by future-example difficulty and stochastic decoding within the constructed meta-training distribution. We directly regress the prospective value of each candidate update against its matched counterfactual target. Since the controller also predicts uncertainty, we use a robust regression term together with a heteroscedastic Gaussian negative log-likelihood:

$$\mathcal{L}_V = \frac{1}{L} \sum_{\ell=1}^{L} \left[\mathrm{Huber}_{\delta_V}\left(\mu_{t,\ell}^{(j)} - \hat{V}_{t,\mathrm{target}}^{(j)}\right) + \frac{\left(\hat{V}_{t,\mathrm{target}}^{(j)} - \mu_{t,\ell}^{(j)}\right)^2}{2(\sigma_{t,\ell}^{(j)})^2} + \log \sigma_{t,\ell}^{(j)} \right]. \tag{11}$$

The counterfactual target is detached from the task model. During meta-training, each per-head term in Eq. (11) is multiplied by its independent Poisson(1) bootstrap weight; gradients update only the prospective value model.

4.3 Update representation and prospective value model

The candidate feature vector $f_t^{(j)}$ describes both the experience and the proposed intervention:

$$f_t^{(j)} = [h_t^{\text{task}}; u_t; g_t^{(j)}; q_t^{(j)}; i_t^{(j)}]. \tag{12}$$

Here h_t^{task} is a pooled hidden representation, u_t contains uncertainty statistics, g_t contains gradient features, q_t summarizes update magnitude and rank, and i_t estimates anchor interference. Using the same fixed sketch map for candidate and protected-anchor gradients, let $\tilde{g}_t^{(j)} = \text{Sketch}(P_j \nabla_w \ell_t)$. We define

$$I_t^{(j)} = \frac{1}{|\mathcal{A}_{\text{prot}}|} \sum_{a \in \mathcal{A}_{\text{prot}}} \max\left(0, -\frac{\langle \tilde{g}_t^{(j)}, g_a \rangle}{\|\tilde{g}_t^{(j)}\| \|g_a\| + \epsilon_{\cos}}\right), \tag{13}$$

where g_a is the frozen anchor-gradient sketch; construction details are given in Appendix C. Unlike replay training, the anchor set is not used to optimize task performance; it is only an input indicating how destructive a write may be.

Q_ψ outputs a mean and scale,

$$(\mu_{t,\ell}^{(j)}, s_{t,\ell}^{(j)}) = Q_{\psi,\ell}\left(f_t^{(j)}\right), \qquad \sigma_{t,\ell}^{(j)} = \text{softplus}(s_{t,\ell}^{(j)}) + \epsilon_\sigma, \tag{14}$$

where $\ell \in \{1, \ldots, L\}$ indexes an ensemble of $L = 4$ heads. The predictive uncertainty used for routing combines the predicted per-head scale with dispersion across ensemble means:

$$(\sigma_t^{(j)})^2 = \frac{1}{L} \sum_{\ell=1}^{L} (\sigma_{t,\ell}^{(j)})^2 + \text{Var}_\ell\left[\mu_{t,\ell}^{(j)}\right], \tag{15}$$

with

$$\mu_t^{(j)} = \frac{1}{L} \sum_{\ell=1}^{L} \mu_{t,\ell}^{(j)}.$$

The resulting uncertainty is not a prediction-confidence score; it quantifies uncertainty about the consequence of learning.

4.4 Lower-confidence-bound write gate and plasticity router

At deployment we compute a conservative score

$$\text{LCB}_t^{(j)} = \mu_t^{(j)} - \beta \sigma_t^{(j)} - \lambda_c C_j - \lambda_i I_t^{(j)}. \tag{16}$$

The no-write action has score zero. The selected action is

$$j_t^* = \arg\max_{j \in \{0, \ldots, M\}} \text{LCB}_t^{(j)}, \qquad z_t = \mathbb{1}[\text{LCB}_t^{(j_t^*)} > \tau_w]. \tag{17}$$

Thus an update is accepted only when its estimated future benefit remains positive after uncertainty, write-cost, and interference penalties. Because the subspace is part of the action, FORESIGHT-TTL can route algebra updates toward late MLP adapters, scientific QA toward attention projections, or decide that no subspace has positive prospective value.

4.5 Reversible fast-weight transactions

Passing the gate does not permanently modify the model. We first create a provisional state

$$\tilde{w}_{t+1} = w_t + z_t \Delta_t^{(j_t^*)} \tag{18}$$

and attach an update record $r_t = (\Delta_t^{(j_t^*)}, j_t^*, \mu_t^{(j_t^*)}, \sigma_t^{(j_t^*)}, t)$.

We monitor each provisional write using subsequent observable weak-reward signals. For the reasoning experiments, let $n_{t,(1)}$ and $n_{t,(2)}$ denote the largest and second-largest answer vote counts among the K sampled trajectories. We use the normalized vote margin

$$\hat{r}_t^+ = \frac{n_{t,(1)} - n_{t,(2)}}{K} \in [0, 1] \tag{19}$$

as the observable weak reward. This is the same self-consistency statistic used by the weak learning objective and requires no ground-truth evaluation label. We additionally use a frozen scalar reference head R_ξ, trained on the no-update branches of the paired meta-training rollouts, to predict the corresponding reward in the absence of the tracked write:

$$\hat{r}^{\mathrm{ref}}_{t+m}(r) = \mathrm{sigmoid}\big(R_\xi(\bar{f}_{t+m,r})\big) \in (0,1). \tag{20}$$

where $\bar{f}_{t+m,r}$ is a fixed-width, record-conditioned feature vector constructed with the same feature layout as $f_t^{(j)}$; its candidate-specific slots are populated from the stored statistics of record r, and no ground-truth evaluation label is included. The reference predictor uses the controller's shared feature representation but stops gradients at the shared trunk; only its separate prediction head is updated by the reference-reward loss. Architecture details are given in Appendix C. It is trained by squared error against the normalized weak reward observed on the matched no-update branch.

For each active provisional record we accumulate discounted excess reward

$$S_{r,k} = \sum_{m=1}^{k} \gamma^{m-1} \left(\hat{r}^{+}_{t+m} - \hat{r}^{\mathrm{ref}}_{t+m}(r)\right), \qquad \gamma = 0.95. \tag{21}$$

We monitor cumulative excess reward with a two-sided conjugate-mixture empirical-Bernstein confidence sequence (Howard et al., 2021), using total error level $\alpha_{\mathrm{CS}} = 0.10$. The exact boundary and implementation details are given in Appendix A.1. A provisional write is committed when the lower confidence bound $L_{r,k}$ on cumulative conditional expected excess reward exceeds $\tau_{\mathrm{commit}} = 1.5$, and is rolled back when the corresponding upper bound satisfies $U_{r,k} < -\tau_{\mathrm{rollback}} = -1.0$. If neither event occurs, the write remains provisional for at most $K_{\max} = 32$ subsequent interactions, after which an uncommitted record is discarded and its stored low-rank delta is removed.

Because each write is stored as an explicit low-rank delta, rollback requires no checkpoint reload and is performed by subtracting the corresponding adapter component.

4.6 Multi-timescale persistence

We separate adaptive state into episodic and persistent components:

$$w_t = w_t^{\mathrm{epi}} + w_t^{\mathrm{pers}}. \tag{22}$$

New accepted writes first enter w^{epi}. After a record is committed, we restart the persistence clock. Let $c(r)$ denote the interaction at which record r is committed. For $k \geq 1$, define the kth post-commit utility observation as

$$\widehat{V}_{r,k} = \hat{r}^{+}_{c(r)+k} - \hat{r}^{\mathrm{ref}}_{c(r)+k}(r). \tag{23}$$

A committed record is eligible for persistent storage after $K_p = 8$ post-commit observations. Its persistence score is

$$P(r) = \frac{\sum_{k=1}^{K_p} \lambda_p^{k-1} \widehat{V}_{r,k}}{\sum_{k=1}^{K_p} \lambda_p^{k-1}}, \qquad K_p = 8, \quad \lambda_p = 0.95. \tag{24}$$

The record is promoted from episodic to persistent state only when

$$P(r) > \tau_p, \qquad \tau_p = 0. \tag{25}$$

Thus promotion requires positive post-commit utility over the default eight-step horizon. Episodic records expire with per-interaction hazard $h_e = 1 - 2^{-1/96}$ (a 96-interaction half-life); persistent records do not decay within an evaluation run and are merged by rank-revealing SVD when the adapter-rank budget is exceeded. The design captures a basic distinction that ordinary TTL ignores: *learning from an experience does not imply remembering it forever*.

Training objective. The prospective utility model Q_ψ is trained solely from matched counterfactual value targets. The reference predictor R_ξ is trained separately on matched no-update weak-reward targets. The base LLM remains frozen during controller meta-training, while candidate LoRA updates are instantiated transiently to construct $\hat{V}^{(j)}_{t,\text{target}}$. We optimize Q_ψ using Eq. (11) for every candidate subspace. Routing requires no separate classification objective: at deployment, candidate values and uncertainties are converted directly into lower-confidence-bound scores by Eq. (16), and Eq. (17) jointly selects the plastic subspace and whether to write.

5 Prospective Gating Analysis

We analyze the behavior of the gate under a simultaneous calibration assumption. Suppose the utility interval is simultaneously calibrated:

$$\mathbb{P}\Big(\forall t, j:\ |\mu_t^{(j)} - V_t(\Delta_t^{(j)})| \le \beta\sigma_t^{(j)}\Big) \ge 1-\delta. \tag{26}$$

Under this simultaneous coverage assumption, the gate would reject candidates with negative true utility on the coverage event for the considered candidate updates. Under this assumption, for the finite candidate set and fixed deployment distribution covered by the calibration procedure,

$$\mathbb{P}(\exists \text{ gate-accepted candidate } j \text{ with } V_t(\Delta_t^{(j)}) < 0) \le \delta. \tag{27}$$

The analysis concerns the *learning action*, not prediction correctness. We therefore measure calibration and harmful-update rate empirically rather than assuming neural intervals are exact. If $|\mu_t^{(j)} - V_t(\Delta_t^{(j)})| \le \epsilon_t$ for all j, the mean-based cost-adjusted selector has one-step regret at most $2\epsilon_t$; the deployed LCB selector additionally includes an uncertainty-dependent conservatism term (Appendix B).

6 Experimental Design

We test whether future-value credit improves online adaptation, retention, calibration, and robustness under noisy test streams. Full data construction, prompts, hyperparameters, baseline recipes, and statistics are in Appendices E–O.

Models and data. We use **Qwen3-8B** (Yang et al., 2025), **DeepSeek-R1-Distill-Qwen-7B** (DeepSeek-AI et al., 2025), and **Llama-3.1-8B-Instruct** (Grattafiori et al., 2024). Counterfactual task streams for controller meta-training draw only from the MATH training split (Hendrycks et al., 2021c) and OpenR1-Math-220k (Lozhkov et al., 2025). A fixed protected anchor set is used to construct interference features and synthetic anti-alignment corruption during controller meta-training, but never serves as a task-learning target or evaluation set. Evaluation uses **AIME 2024/2025** (Zhang & Math-AI Team, 2024; 2025), **MATH-500** (Lightman et al., 2024), **GPQA-Diamond** (Rein et al., 2024), **BBH** (Suzgun et al., 2023), and the 120 public evaluation tasks of **ARC-AGI-2** (Chollet et al., 2025). The held-out competition-mathematics block in ReasonStream-1K uses public MathArena problem releases (Balunovic et al., 2025; Dekoninck et al., 2026).

Streaming protocol. ReasonStream-1K is a 1,000-example non-IID stream with five 200-step blocks: MATH algebra, MATH geometry, GPQA science, five BBH logic tasks, and held-out competition mathematics. For each stream seed, the sampled examples and within-block order are fixed before any method is run. Read-only held-out retention probes appear every 50 steps and never alter adaptive state. Candidate writes are rank-16 LoRA deltas over six attention/MLP/episodic subspaces; slow backbone and safety-critical output weights remain frozen.

Baselines and metrics. We compare Static inference, TTRL (Zuo et al., 2025), DDRL (Yu et al., 2026), TTSR (He et al., 2026), and EvoLib (Xu et al., 2026) under identical endpoint evaluation protocols. Parameter-updating baselines use the common rank-16 LoRA capacity and meta-validation budget described in Appendix G, while retaining their method-specific adaptation rules. Besides final accuracy, we report future update utility $\text{FUU} = 100\mathbb{E}[V_t \mid z_t = 1]$, harmful-update rate $\text{HUR} = \mathbb{P}(V_t < 0 \mid z_t = 1)$, update precision $\text{UP} = \mathbb{P}(V_t > 0 \mid z_t = 1)$, and retention on held-out read-only probes. Corruption tests replace 0–40% of learning experiences with misleading pseudo-labels or perturbed traces without exposing corruption indicators to the learner.

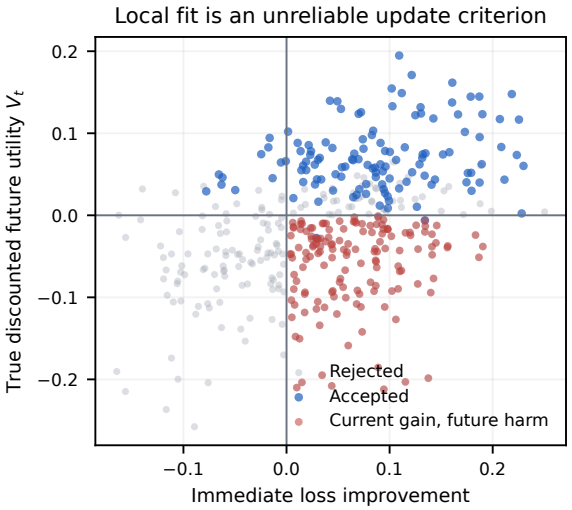

Figure 2: **Immediate optimization and future utility can disagree.** Each point is a candidate update. Red points improve the source objective but hurt the future stream. FORESIGHT-TTL accepts only candidates whose prospective lower confidence bound is positive.

7 RESULTS AND ANALYSIS

All quantitative analyses in this section are based on the experiments described in Section 6 and the appendices.

7.1 IMMEDIATE FIT IS NOT FUTURE UTILITY

Figure 2 visualizes the router-selected candidate at 420 held-out decision points. Across five stream seeds, we report mean values. The key failure mode is not rare: **35.2%** of these candidates lie in the lower-right quadrant, where the triggering objective improves but discounted future utility is negative. This is exactly the regime in which conventional TTL has no principled reason to abstain. In contrast, the LCB gate writes at 29.0% of these decision points, and only 1.6% of realized writes have negative measured future utility. Accepted updates have mean FUU $= +7.05$ points (i.e., $\mathbb{E}[V_t \mid z_t = 1] = 0.0705$), whereas the unconditional candidate pool has mean future utility of -1.33 points (i.e., $\mathbb{E}[V_t] = -0.0133$). The result provides empirical evidence that selective plasticity can improve over update-only optimization in this evaluation setting.

7.2 PROSPECTIVE SELECTION IMPROVES STREAM STABILITY

Figure 3 compares online accuracy on REASONSTREAM-1K, averaged over five stream-order and decoding seeds. FORESIGHT-TTL reaches 65.7% mean online accuracy versus 61.1% for both DDRL and EvoLib, and ends the final competition block at 59.0% versus 51.6%/52.4%. The gap widens after repeated domain switches.

7.3 UPDATE VALUE IS CALIBRATED AND ROBUST TO CORRUPTED STREAMS

Figure 4 separates two questions. First, the value model is well calibrated across negative and positive updates: estimated and observed future utility follow the diagonal rather than collapsing to a binary confidence heuristic. Second, calibration translates into robust abstention. At 20% corrupted experiences, the harmful-update rate rises to 44% for TTRL and 30% for DDRL, but remains 8.9% for FORESIGHT-TTL. Even at 40% corruption, only 15.5% of accepted writes have negative measured future utility, less than one quarter of TTRL's rate. The mechanism is not that FORESIGHT-TTL detects corruption directly; it learns that candidate updates with unstable pseudo-labels, high interference, and uncertain downstream consequences have low lower-confidence-bound value.

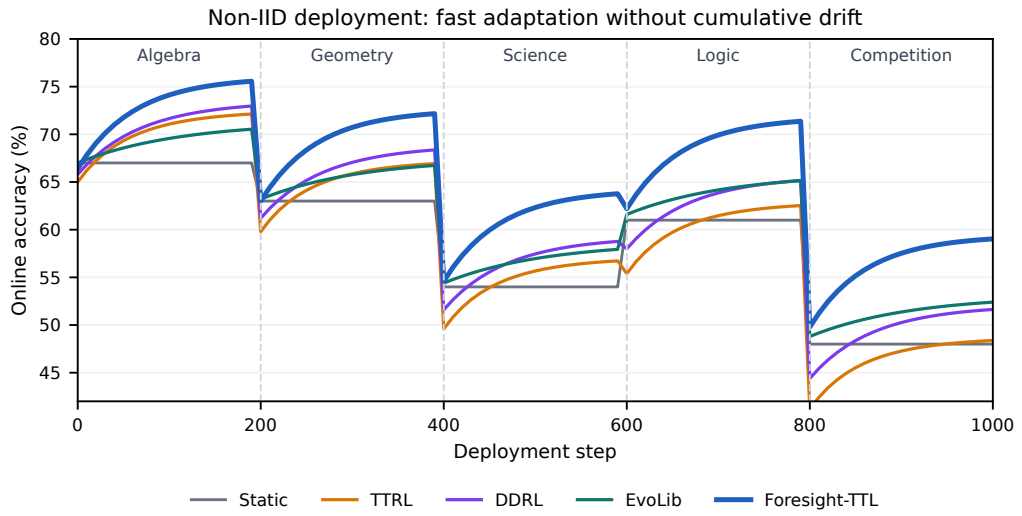

Figure 3: **Online performance under five non-IID domain blocks.** Vertical dashed lines denote distribution shifts. Prospective gating reduces the oscillation and cumulative drift that arise when every locally plausible update is retained.

Table 1: **Final-answer accuracy (%) after test-time adaptation on DeepSeek-R1-Distill-Qwen-7B.** The table reports endpoint task quality; dynamic stream behavior, update safety, and calibration are shown separately in Figures 3–4. Best and second-best results are bolded and underlined respectively for each column.

Method	AIME'24	AIME'25	MATH-500	GPQA-D	BBH	ARC2-p@1
Static	53.8	44.7	89.8	48.2	78.3	4.1
TTRL	61.1	50.0	91.9	49.8	79.8	4.8
DDRL	<u>62.2</u>	51.3	93.9	50.6	80.3	5.2
TTSR	61.7	<u>52.0</u>	<u>94.1</u>	<u>51.1</u>	80.5	5.7
EvoLib	59.4	49.3	93.5	50.4	<u>80.8</u>	<u>6.0</u>
FORESIGHT-TTL	**64.4**	**55.3**	**94.7**	**52.8**	**82.1**	**7.5**

7.4 ENDPOINT ACCURACY AND CROSS-DOMAIN GENERALIZATION

Table 1 reports final-answer accuracy under a common DeepSeek-R1-Distill-Qwen-7B protocol (Appendix F). FORESIGHT-TTL reaches 64.4/55.3 on AIME'24/'25 versus 62.2/51.3 for DDRL, while gains are smaller on the near-saturated MATH-500 benchmark. ARC-AGI-2 scores remain low for all 7B methods, but selective adaptation yields the largest improvement in this evaluation.

Adaptation versus retention. FORESIGHT-TTL gains +9.4 points within-domain with only -0.8 held-out retention change, versus +6.9/-3.8 for DDRL and +5.8/-5.7 for TTRL; full ablations are in Appendix L. Figure 7 visualizes the full frontier.

Compute cost. FORESIGHT-TTL runs at $1.16\times$ static wall-clock versus $1.38\times$ for TTRL and $1.31\times$ for DDRL; no counterfactual branching is required on the live deployment path. Paired branches are used only offline for controller meta-training and evaluation-only utility audits (Appendix M).

8 DISCUSSION AND CONCLUSION

Prospective credit separates answer confidence from whether an update improves future decisions. FORESIGHT-TTL combines value-based gating, reversible writes, and persistence, but depends on measurable meta-training futures, a finite horizon H, and explicitly tracked adaptive state.

Reproducibility Statement

Appendices provide the complete algorithm, proof details, stream construction, evaluation protocol, hyperparameters, baseline configurations, hardware accounting, statistical tests, and additional ablations.

Ethics Statement

TTL changes a deployed model using its own interactions, which creates risks absent from static inference: malicious inputs may attempt to poison adaptive state, and benign but misleading experiences can induce persistent regressions. Our method is explicitly motivated by these risks. We recommend sandboxing provisional weights, maintaining immutable safety-critical parameters, logging update provenance, restricting promotion to persistent state, and evaluating adversarial streams before deployment. The experiments use public benchmarks and require no human-subject data.

AI Use Statement

LLMs are utilized in this manuscript for partial grammatical checks and language polishing. The authors are fully responsible for the final content.

References

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/akyurek25a.html.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Yishuo Cai, Xingyu Guo, Xuancheng Huang, Jinhua Du, Can Huang, Wenxuan Huang, Wenhan Ma, Yuyang Hu, Aohan Zeng, Jie Tang, and Xu Sun. From player to master: Enhancing test-time learning of LLM agents via reinforcement learning over memory. *arXiv preprint arXiv:2606.08656*, 2026. URL https://arxiv.org/abs/2606.08656.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1179c561a4efc77fbfda76aa1addce33-Abstract-Conference.html.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025. URL https://arxiv.org/abs/2505.11831.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvalddson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs. *arXiv preprint arXiv:2605.00674*, 2026. URL https://arxiv.org/abs/2605.00674.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ba6f5879d7f382241316f7dda7e6b8ac-Abstract-Conference.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Haoyang He, Zihua Rong, Yunjia Zhao, Lan Yang, Jian Chang, and Honggang Zhang. TTSR: Test-time self-evolving via reflection. *arXiv preprint arXiv:2603.03297*, 2026. URL https://arxiv.org/abs/2603.03297.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning AI with shared human values. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=dNy_RKzJacY.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021c. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.

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.

Vanshaj Khattar, Md. Rafi Ur Rashid, Moumita Choudhury, Jing Liu, Toshiaki Koike-Akino, Ming Jin, and Ye Wang. Amplification effects in test-time reinforcement learning: Safety and reasoning vulnerabilities. *arXiv preprint arXiv:2603.15417*, 2026. URL https://arxiv.org/abs/2603.15417.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Guilherme Penedo, Edward Beeching, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. OpenR1-Math-220k. Hugging Face repository, 2025. URL https://huggingface.co/datasets/open-r1/OpenR1-Math-220k.

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.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

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 *International Conference on Machine Learning*, 2020.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

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.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022.

Weijia Xu, Alessandro Sordoni, Chandan Singh, Zelalem Gero, Michel Galley, Xingdi Yuan, and Jianfeng Gao. Test-time learning with an evolving library. *arXiv preprint arXiv:2605.14477*, 2026. URL https://arxiv.org/abs/2605.14477.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. TTCS: Test-time curriculum synthesis for self-evolving. In *Lifelong Agents Workshop at ICLR*, 2026. URL https://openreview.net/pdf?id=ZkAJQNXQkr.

Yongcan Yu, Lingxiao He, Jian Liang, Kuangpu Guo, Meng Wang, Qianlong Xie, Xingxing Wang, and Ran He. Understanding and mitigating spurious signal amplification in test-time reinforcement learning for math reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37424–37436, 2026. URL https://aclanthology.org/2026.findings-acl.1865/.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William Freeman, and Hao Tan. Test-time training done right. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ffbfff81fb78e4bb558273b91e12e318-Abstract-Conference.html.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2024. Hugging Face Dataset, 2024.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Cai Zhou, Zekai Wang, Menghua Wu, Qianyu Julie Zhu, Flora C. Shi, Chenyu Wang, Ashia Wilson, Tommi Jaakkola, and Stephen Bates. Online reasoning calibration: Test-time training enables generalizable conformal LLM reasoning. *arXiv preprint arXiv:2604.01170*, 2026. URL https://arxiv.org/abs/2604.01170.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4376. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/be690ea16f005c174f6c4102a5970e67-Abstract-Conference.html.

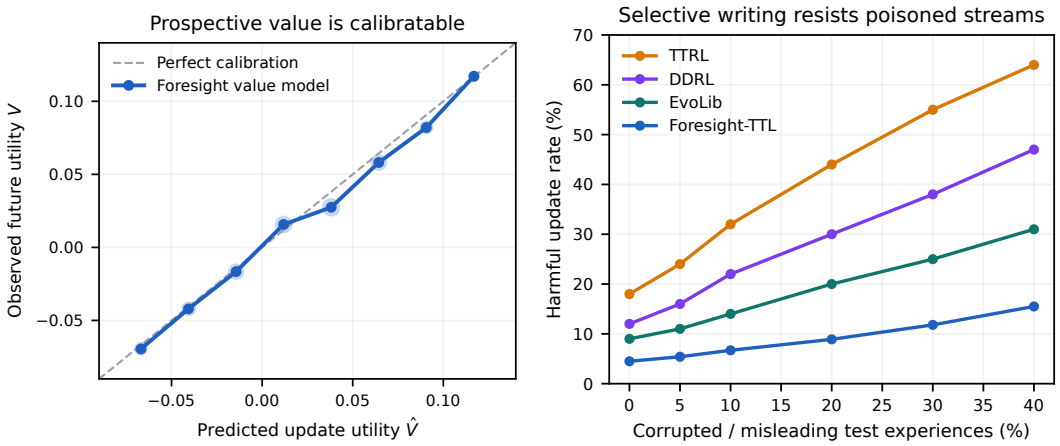

Figure 4: **Prospective calibration and robustness.** Left: binned estimated update utility versus held-out offline audit utility $V^{(j)}_{t,\mathrm{audit}}$. Right: harmful-update rate as misleading experiences are injected into the learning stream.

A Full Algorithm

Algorithm 1 gives the meta-training procedure and Algorithm 2 gives deployment. The key engineering choice is that counterfactual branches are never required on the live deployment path. Paired branches are used offline during controller meta-training and for evaluation-only utility audits. Deployment itself maintains a single task-model state plus low-rank update records.

Algorithm 1 Meta-training the prospective update-value controller

Require: Frozen backbone θ, stream generator $\mathcal{S}$, candidate subspaces $\{S_j\}_{j=1}^{M}$, counterfactual horizon H

initialize prospective value model Q_ψ, reference head R_ξ, and protected anchor sketch $\mathcal{A}_{\mathrm{prot}}$
for stream prefix $(e_{1:t}, B_{t+1:t+H}) \sim \mathcal{S}$ **do**
  solve e_t and construct weak loss ℓ_t using Eq. (7)
  for $j = 1, \dots, M$ **do**
    propose $\Delta_t^{(j)} = -\eta_j P_j \nabla \ell_t$
    clone matched update/baseline adaptive states $w_t^+ = w_t + \Delta_t^{(j)}$, $w_t^0 = w_t$
    evaluate both branches on identical $B_{t+1:t+H}$ and seeds, with no additional writes inside the counterfactual window
    compute target $\hat{V}^{(j)}_{t,\mathrm{target}} = G_{t,H}(w_t^+) - G_{t,H}(w_t^0)$
    construct feature vector $f_t^{(j)}$ including Eq. (13)
  end for
  fit Q_ψ on all candidate utility targets using Eq. (11)
  fit R_ξ to per-step weak rewards from the matched no-update branches
end for
calibrate ensemble uncertainty on meta-validation streams and freeze Q_ψ and R_ξ

A.1 Sequential confidence boundary

For each provisional record r, define the discounted observable excess-reward increment

$$X_{r,m} = \gamma^{m-1}\left(\hat{r}^{+}_{t+m} - \hat{r}^{\mathrm{ref}}_{t+m}(r)\right). \tag{28}$$

Because both weak-reward terms lie in $[0,1]$ and $\gamma \le 1$, we have $X_{r,m} \in [-1,1]$.

Algorithm 2 FORESIGHT-TTL at deployment

Require: Frozen backbone θ, episodic/persistent adapters $(w^{\mathrm{epi}}, w^{\mathrm{pers}})$, prospective value model Q_ψ, and frozen reference head R_ξ

for $t = 1, 2, \ldots$ **do**
  generate the task response and the K learning trajectories using π_{θ, w_t}
  compute the observable weak reward $\hat{r}_t^+$
  for each provisional record r created before interaction t **do**
    compute $\hat{r}_t^{\mathrm{ref}}(r) = \mathrm{sigmoid}\big(R_\xi(\bar{f}_{t,r})\big)$ and update $S_{r,k}$ using Eq. (21)
    update $(L_{r,k}, U_{r,k})$ using Eq. (32)
    if $L_{r,k} > \tau_{\mathrm{commit}}$ **then**
      commit r
    else if $U_{r,k} < -\tau_{\mathrm{rollback}}$ **then**
      subtract Δ_r and delete r
    else if $k = K_{\max}$ **then**
      subtract Δ_r and delete r
    end if
  end for
  update post-commit persistence scores for records committed before interaction t
  for each episodic record r satisfying Eq. (24) **do**
    move Δ_r from w^{epi} to w^{pers} without changing the total adaptive state w_t, and mark r persistent
  end for
  independently expire each committed episodic record with probability $h_e = 1 - 2^{-1/96}$; upon expiry, subtract its stored delta and delete the record
  if the persistent adapter-rank budget exceeds 128 **then**
    compress the persistent adapter state by rank-revealing SVD to the rank-128 budget
  end if
  form the weak learning signal and candidate updates $\{\Delta_t^{(j)}\}$
  score candidates with Eq. (16) and obtain (j_t^*, z_t) using Eq. (17)
  if $z_t = 1$ **then**
    add $\Delta_t^{(j_t^*)}$ to episodic state and create provisional record r_t
  end if
end for

Let

$$S_{r,k} = \sum_{m=1}^{k} X_{r,m} \tag{29}$$

and define the predictable centering sequence

$$\widehat{X}_{r,1} = 0, \qquad \widehat{X}_{r,m} = \frac{1}{m-1} \sum_{q=1}^{m-1} X_{r,q}, \quad m > 1. \tag{30}$$

The empirical intrinsic-time process is

$$\widehat{\mathcal{V}}_{r,k} = \sum_{m=1}^{k} \left(X_{r,m} - \widehat{X}_{r,m} \right)^2. \tag{31}$$

We use the conjugate-mixture empirical-Bernstein confidence sequence of Howard et al. (2021). Since $X_{r,m} \in [-1, 1]$, the sub-exponential scale is $c = 2$. Let $u_{\mathrm{GE}}(v; \alpha, c, v_{\mathrm{opt}})$ denote the gamma-exponential mixture uniform boundary, with its mixture parameter chosen to optimize the boundary at intrinsic time v_{opt}. We use

$$\alpha_{\mathrm{CS}} = 0.10, \qquad \alpha_{\mathrm{tail}} = \alpha_{\mathrm{CS}}/2 = 0.05, \qquad v_{\mathrm{opt}} = K_{\max} = 32.$$

The resulting confidence sequence for the cumulative conditional expected excess reward

$$M_{r,k} = \sum_{m=1}^{k} \mathbb{E}_{m-1}[X_{r,m}]$$

is

$$L_{r,k} = S_{r,k} - u_{\mathrm{GE}}\left(\widehat{\mathcal{V}}_{r,k}; 0.05, 2, 32\right), \qquad U_{r,k} = S_{r,k} + u_{\mathrm{GE}}\left(\widehat{\mathcal{V}}_{r,k}; 0.05, 2, 32\right). \tag{32}$$

The gamma-exponential boundary is evaluated by numerical root finding, following the reference implementation of Howard et al. (2021). The same confidence-sequence parameters are used for every benchmark and backbone. A provisional record is committed when $L_{r,k} > \tau_{\mathrm{commit}}$ and rolled back when $U_{r,k} < -\tau_{\mathrm{rollback}}$.

B Theoretical Details

B.1 Harmful-acceptance proposition

Proposition 1. Suppose that for all candidate updates considered over a finite deployment horizon, the simultaneous coverage event

$$\mathcal{E} = \bigcap_{t,j} \left\{ |\mu_t^{(j)} - V_t^{(j)}| \le \beta \sigma_t^{(j)} \right\}$$

has probability at least $1 - \delta$. Suppose costs $C_j, I_j \ge 0$ and the write gate accepts a candidate only if $\mu_t^{(j)} - \beta\sigma_t^{(j)} - \lambda_c C_j - \lambda_i I_j > 0$. Then the probability of accepting any candidate with $V_t^{(j)} < 0$ is at most δ.

Proof. Condition on $\mathcal{E}$. For an accepted candidate,

$$V_t^{(j)} \ge \mu_t^{(j)} - \beta\sigma_t^{(j)} > \lambda_c C_j + \lambda_i I_j \ge 0.$$

Thus no harmful candidate is accepted on $\mathcal{E}$. The complement has probability at most δ. □

The proposition deliberately assumes simultaneous calibration, which a learned neural interval does not automatically provide. Our empirical protocol therefore includes held-out calibration, reliability plots, and direct harmful-update measurement. In a deployment requiring stronger guarantees, the learned intervals can be wrapped by conformal calibration or time-uniform confidence sequences at the cost of more conservative writing.

B.2 One-step selector regret

Define the cost-adjusted penalty

$$c_t^{(j)} = \lambda_c C_j + \lambda_i I_t^{(j)}.$$

Define net candidate utility $U_t^{(j)} = V_t^{(j)} - c_t^{(j)}$ and estimated utility $\hat{U}_t^{(j)} = \mu_t^{(j)} - c_t^{(j)}$. Include the no-write action $j = 0$ with $U_t^{(0)} = \hat{U}_t^{(0)} = 0$. Let

$$j^* = \arg\max_{j \in \{0,\dots,M\}} U_t^{(j)}, \qquad \hat{j} = \arg\max_{j \in \{0,\dots,M\}} \hat{U}_t^{(j)}.$$

If $|\hat{U}_t^{(j)} - U_t^{(j)}| \le \epsilon_t$ for all j, then

$$U_t^{(j^*)} - U_t^{(\hat{j})} \le 2\epsilon_t.$$

This follows from $U(j^*) \le \hat{U}(j^*) + \epsilon_t \le \hat{U}(\hat{j}) + \epsilon_t \le U(\hat{j}) + 2\epsilon_t$. An LCB selector replaces $\hat{U}$ by $\hat{U} - \beta\sigma$, adding a conservatism term but reducing harmful selection when interval uncertainty is informative.

B.3 Rollback loss

If a harmful provisional write has per-step utility lower bounded by $-B$ and is rolled back within K_r interactions, its worst-case extra regret is BK_r. With multiple non-overlapping tracked writes, $\mathcal{R}_{\mathrm{rollback}} \le B \sum_r K_r$. This makes rollback speed a measurable property rather than a qualitative safety claim.

C Update Representation and Feature Construction

Controller architecture. The prospective controller maps each candidate feature vector $f_t^{(j)}$ to a 1024-dimensional hidden representation with a learned linear projection. The shared trunk contains three residual feed-forward blocks. Each block applies RMSNorm followed by a $1024 \to 2048 \to 1024$ MLP with SiLU activation and a residual connection.

The uncertainty ensemble contains four independently initialized heads $\{Q_{\psi,\ell}\}_{\ell=1}^{4}$ sharing this trunk. Each head consists of a $1024 \to 256 \to 2$ MLP with SiLU activation and outputs $(\mu_{t,\ell}^{(j)}, s_{t,\ell}^{(j)})$, where $\sigma_{t,\ell}^{(j)} = \mathrm{softplus}(s_{t,\ell}^{(j)}) + \epsilon_\sigma$. During meta-training, each head receives an independent $\mathrm{Poisson}(1)$ bootstrap weight for every candidate-value training record, in addition to independent parameter initialization. This induces ensemble diversity without stochastic inference-time dropout.

The reference predictor R_ξ receives the same 1024-dimensional shared trunk representation of the fixed-width record-conditioned feature vector $\bar{f}_{t,r}$. Gradients from its loss are stopped at the shared trunk. Its separate $1024 \to 256 \to 1$ SiLU MLP is trained on matched no-update weak-reward targets, with a sigmoid output constraining $\hat{r}^{\mathrm{ref}}$ to $(0, 1)$. The complete controller, including the four utility heads and reference head, contains approximately 18M trainable parameters.

Plastic subspaces. We attach LoRA (Hu et al., 2022) modules only to adaptive layers. For Qwen3-8B and DeepSeek-R1-Distill-Qwen-7B, the six candidate groups are: (1) query/value projections in the upper 12 blocks, (2) attention output projections in the upper 12 blocks, (3) MLP up/gate projections, (4) MLP down projections, (5) all LoRA modules in the final four blocks as a joint subspace, and (6) a separate episodic adapter bank. Each candidate uses rank 16, $\alpha_{\mathrm{LoRA}} = 32$, dropout 0, and a maximum Frobenius-norm update of 0.15 after normalization. The Llama-3.1 configuration uses the analogous projection names. The corresponding candidate learning rates for groups (1)–(6) are $(5, 5, 8, 5, 8, 10) \times 10^{-5}$, respectively.

Gradient features. To avoid feeding millions of gradient coordinates to Q_ψ, we compute: global ℓ_2 norm; per-module norm quantiles; fraction of norm in the top four layers; cosine similarity to the previous accepted update; cosine similarity to a running exponential gradient average; the stable rank of the low-rank delta; and a 64-dimensional CountSketch of the flattened gradient. The same fixed CountSketch map is used for candidate gradients and protected anchor gradients in Eq. (13). These are concatenated with the pooled hidden state from the final prompt token.

Uncertainty features. For $K = 8$ sampled reasoning trajectories we record answer entropy, majority fraction, normalized vote margin, average token entropy over the final 25% of each trajectory, and disagreement in normalized solution length. These features intentionally include but do not rely on self-consistency. The utility model can learn that a high-consistency update is valuable in one domain yet dangerous in another when its gradient conflicts with anchors.

Anchor sketch. The protected anchor set $\mathcal{A}_{\mathrm{prot}}$ contains 128 fixed prompts selected before all experiments: 32 general-knowledge, 32 instruction-following, 32 arithmetic, and 32 safety/integrity prompts. Protected-anchor inputs use the same source-specific rendering rules as the corresponding retention examples described in Appendix C.1. No gold answer or evaluator output is included in the anchor input; the preservation target is solely the frozen reference completion y_a^{ref} generated by the pre-adaptation model. For each anchor prompt a, the frozen pre-adaptation model generates a deterministic reference completion y_a^{ref}. Reference completions are generated once using greedy decoding (temperature 0) with a maximum generation length of 1,024 tokens and are then frozen for all controller training and evaluation runs. We define the anchor-preservation loss as the mean token-level negative log-likelihood of this frozen reference completion,

$$\ell_{\mathrm{prot}}(a; w) = -\frac{1}{|y_a^{\mathrm{ref}}|} \sum_{s=1}^{|y_a^{\mathrm{ref}}|} \log p_{\theta,w}\big(y_{a,s}^{\mathrm{ref}} \mid a, y_{a,<s}^{\mathrm{ref}}\big) . \tag{33}$$

Before deployment, we compute and freeze the corresponding low-dimensional gradient sketch

$$g_a = \mathrm{Sketch}\big(\nabla_w \ell_{\mathrm{prot}}(a; w)\big|_{w=0}\big) . \tag{34}$$

Thus, a candidate gradient that is anti-aligned with g_a tends to increase the loss of the reference behavior and receives a larger interference penalty in Eq. (13). We store only the low-dimensional gradient sketches, frozen reference completions, and reference logits; the anchor set is never optimized against at deployment.

C.1 Fixed protected and retention sets

We construct the protected and retention sets from four public benchmark releases: MMLU test (Hendrycks et al., 2021b) for general knowledge, IFEval (Zhou et al., 2023) for instruction following, GSM8K test (Cobbe et al., 2021) for arithmetic, and the ETHICS commonsense test split (Hendrycks et al., 2021a) for safety/integrity. We use the benchmark releases accompanying the cited works.

After removing any normalized-text match with controller meta-training data or endpoint benchmarks, examples are stratified as follows. MMLU is divided into four broad subject groups: humanities, social sciences, STEM, and other. We fix the subject-to-group mapping explicitly in Table 2, following the grouping used by the standard MMLU evaluation implementation. IFEval is divided into four strata: single-constraint prompts whose whitespace-token length is at or below the median among all single-constraint prompts, single-constraint prompts whose length is strictly above that median, two-constraint prompts, and three-constraint prompts. The median is computed once on the pinned IFEval revision before any protected or retention examples are selected. GSM8K is divided into quartiles according to the number of whitespace-separated tokens in the reference solution, and ETHICS commonsense is stratified by its binary target label.

Table 2: Fixed MMLU subject-to-category mapping used for protected and retention-set construction.

Subject	Group
abstract_algebra	STEM
anatomy	Other
astronomy	STEM
business_ethics	Other
clinical_knowledge	Other
college_biology	STEM
college_chemistry	STEM
college_computer_science	STEM
college_mathematics	STEM
college_medicine	Other
college_physics	STEM
computer_security	STEM
conceptual_physics	STEM
econometrics	Social sciences
electrical_engineering	STEM
elementary_mathematics	STEM
formal_logic	Humanities
global_facts	Other
high_school_biology	STEM
high_school_chemistry	STEM
high_school_computer_science	STEM
high_school_european_history	Humanities
high_school_geography	Social sciences
high_school_government_and_politics	Social sciences
high_school_macroeconomics	Social sciences
high_school_mathematics	STEM
high_school_microeconomics	Social sciences
high_school_physics	STEM
high_school_psychology	Social sciences
high_school_statistics	STEM
high_school_us_history	Humanities
high_school_world_history	Humanities

Subject	Group
human_aging	Other
human_sexuality	Social sciences
international_law	Humanities
jurisprudence	Humanities
logical_fallacies	Humanities
machine_learning	STEM
management	Other
marketing	Other
medical_genetics	Other
miscellaneous	Other
moral_disputes	Humanities
moral_scenarios	Humanities
nutrition	Other
philosophy	Humanities
prehistory	Humanities
professional_accounting	Other
professional_law	Humanities
professional_medicine	Other
professional_psychology	Social sciences
public_relations	Social sciences
security_studies	Social sciences
sociology	Social sciences
us_foreign_policy	Social sciences
virology	Other
world_religions	Humanities

Within every stratum, examples are deterministically ordered by

$$h(x) = \mathrm{SHA256}(\texttt{SP-v1} \,\|\, \mathrm{source}(x) \,\|\, \mathrm{normalize}(\mathrm{serialize}(x))) \,, \tag{35}$$

using ascending hexadecimal hash order, where $\mathrm{serialize}(x)$ is the canonical benchmark instance containing the prompt, answer options or constraints when applicable, and reference target. The normalization operator applies Unicode NFC normalization, converts all line endings to "\n", strips leading and trailing whitespace from each serialized field, and collapses each run of whitespace characters to a single ASCII space. Case, punctuation, numbers, mathematical symbols, and answer-option order are otherwise left unchanged. SHA256 is computed over the resulting UTF-8 byte string.

For MMLU, IFEval, and GSM8K, the first 8 examples in each stratum are assigned to $\mathcal{A}_{\mathrm{prot}}$ and the next 20 to $\mathcal{A}_{\mathrm{ret}}$. For ETHICS commonsense, the first 16 examples from each label are assigned to $\mathcal{A}_{\mathrm{prot}}$ and the next 40 from each label to $\mathcal{A}_{\mathrm{ret}}$.

Consequently,

$$|\mathcal{A}_{\mathrm{prot}}| = 4 \times 32 = 128, \qquad |\mathcal{A}_{\mathrm{ret}}| = 4 \times 80 = 320.$$

The two sets are mutually disjoint and are fixed before controller meta-training, hyperparameter selection, and benchmark evaluation.

Retention scoring follows the native deterministic evaluator of each source: exact multiple-choice accuracy for MMLU, strict instruction-following success for IFEval, exact final numeric answer for GSM8K, and binary classification accuracy for ETHICS commonsense. The four category accuracies are macro-averaged so that each capability contributes equally to the reported retention score.

All retention examples are evaluated zero-shot with no in-context demonstrations. MMLU and ETHICS commonsense are rendered as multiple-choice questions with explicitly labeled answer options, and the first valid option label in the model's final answer is scored. IFEval uses the original benchmark prompt without an additional task instruction and is scored with the official strict evaluator. GSM8K uses the same mathematical-reasoning system instruction as the main reasoning experiments; the final numeric answer is extracted from the last valid boxed answer, falling back to the final parseable number. The same formatting and extraction rules are used under w_0 and w_{end}.

The exact Hugging Face dataset revisions used to construct the fixed protected and retention sets are: `cais/mmlu@c30699e8356da336a370243923dbaf21066bb9fe`, `google/IFEval@39ed06ce3906b51290c6e95b7c697e928c8a7b00`, `openai/gsm8k@a8402b897b382048aed4a75738da3c83e908a7f9`, and `hendrycks/ethics@b8b47c589f8bee77175b8648e5497278b68da48a`. All examples are selected from these immutable revisions before any meta-training, hyperparameter selection, or benchmark evaluation.

D Why Paired Counterfactual Rollouts

A naive target compares future performance after an update with an independently sampled no-update rollout. If future example difficulty has variance σ_B^2, the variance of the difference contains roughly $2\sigma_B^2$. Matched future examples cancel this common variation. The same principle is used in paired benchmarking: the comparison is between applying the update and skipping the update, rather than between independently sampled future batches.

In the paired-rollout experiment, paired targets reduce the standard deviation of utility estimates from 5.8 to 2.6 reward points at $H = 8$. Both branches use identical generation seeds where deterministic kernels permit; otherwise they share per-example random-number-generator states at branch creation.

E Meta-Training Streams

Before constructing meta-training streams, we remove evaluation contamination from the source corpora by normalized problem-text matching. The exclusion set is the union of all endpoint evaluation instances and all examples eligible for ReasonStream-1K, including AIME 2024/2025, MATH-500, the MATH test split, GPQA and GPQA-Diamond, the BBH tasks used for streaming and transfer evaluation, ARC-AGI-2, and the held-out competition-mathematics pool.

For matching, we canonicalize Unicode, whitespace, and non-semantic LaTeX formatting before exact comparison while preserving numerical and symbolic content. Any meta-training example whose normalized problem statement matches an instance in the exclusion set is removed before stream construction.

After filtering, 60% of source questions are drawn from the MATH training split and 40% from OpenR1-Math-220k. A stream episode contains 64–192 problems. Topic distributions follow a piecewise-stationary Dirichlet process: every 16–48 steps a new mixture over algebra, counting/probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus is sampled. Difficulty is estimated using source metadata and baseline solution length.

To train abstention, 35% of stream episodes contain synthetic corruption. Corruption is applied at individual learning positions. At each corrupted position, one of four applicable corruption operators is sampled uniformly: (i) replace the majority pseudo-label by the runner-up answer; (ii) replace the final answer of a trajectory from a non-modal answer cluster by the modal pseudo-answer while leaving its incompatible reasoning trace unchanged; (iii) perturb one key numerical quantity in the copy of the prompt used only to construct the weak learning signal, while leaving the externally scored task input unchanged; or (iv) replace the weak learning signal by an anti-aligned distractor trajectory.

Operators (i) and (ii) are applicable only when at least two distinct answer clusters are present. If a sampled operator is inapplicable, we resample uniformly from the remaining applicable operators using the same stream-seed random generator. Every selected corruption position therefore receives exactly one corruption.

For operator (iv), we construct a fixed corruption bank $\mathcal{B}_{\mathrm{corr}}$ from the decontaminated meta-training source pool before controller training. For each protected-anchor category c, define its normalized gradient-sketch centroid

$$\bar{g}_c = \frac{\sum_{a \in \mathcal{A}_c} g_a / \|g_a\|}{\left\| \sum_{a \in \mathcal{A}_c} g_a / \|g_a\| \right\|}. \tag{36}$$

For every candidate example x in the corruption bank, we compute

$$g_{\mathrm{corr}}(x) = \mathrm{Sketch}\left(\nabla_w \ell_{\mathrm{weak}}(x; w)\big|_{w=0}\right), \tag{37}$$

where ℓ_{weak} is Eq. (7), using the same $K = 8$ trajectory-generation configuration as deployment. Candidate examples in $\mathcal{B}_{\mathrm{corr}}$ are ranked by the cosine similarity between $g_{\mathrm{corr}}(x)$ and $\bar{g}_c$. An anti-aligned distractor is sampled uniformly from the bottom 10% for a uniformly sampled anchor category. Only its weak learning trajectory and pseudo-label replace the learning signal at the corrupted position; the original task input and external evaluation label remain unchanged.

The corruption bank is disjoint from all endpoint benchmarks, $\mathcal{A}_{\mathrm{prot}}$, and $\mathcal{A}_{\mathrm{ret}}$. Corruption indicators are never exposed to the learner or controller.

Counterfactual reward. For all reasoning experiments, the reward R in Eq. (4) is the per-example task score in $[0, 1]$. For mathematical free-response tasks this is exact final-answer correctness; for multiple-choice tasks it is answer correctness, and for ARC-style tasks it is exact grid match. During controller meta-training, ground-truth answers are used only offline to construct matched counterfactual value targets and are never exposed to the deployment-time learner or controller. Consequently, $\hat{V}^{(j)}_{t,\mathrm{target}}$ measures the discounted difference in future task correctness between applying and skipping candidate j.

For the main model, all counterfactual targets use a fixed horizon $H = 8$. We generate 40k meta-training batches of four stream prefixes, corresponding to 160k stream prefixes in total. Each prefix produces up to $M = 6$ candidate intervention records, yielding up to 960k candidate value targets before filtering degenerate zero-gradient candidates. Generated records are optimized with AdamW using minibatches of 128 update records, learning rate 3×10^{-4}, weight decay 0.01, and gradient clipping at 1.0.

F Evaluation Stream Construction

Unless otherwise stated, endpoint benchmarks are evaluated prequentially. Adaptive state is reset at the beginning of each benchmark and each evaluation seed. For each example, all responses contributing to reported accuracy are generated and scored before any update derived from that example is applied. The learner may then use only the prompt, sampled trajectories, and weak test-time signals to construct an update that can affect subsequent examples. Ground-truth evaluation labels are never exposed to the learner. ARC-AGI-2 follows the separate per-task demonstration protocol described below. Within each endpoint benchmark, the example order is fixed before any method is run and is shared identically across all compared methods for the same evaluation seed. The ordering is determined solely by the evaluation seed and is never adapted to model outputs or intermediate performance.

Offline audit of update utility. For metric computation, an *adaptation event* is any state-changing update actually applied by the evaluated method. For FORESIGHT-TTL, this is a gate-accepted LoRA write. For TTRL, DDRL, and TTSR, it is an optimizer update applied to the adaptive LoRA parameters. For EvoLib, it is a modification of the deployed textual library. We write $z_t^{(m)} = 1$ when method m applies such an event.

For every realized adaptation event, we construct a matched offline skip-event audit from the same pre-event adaptive state. One branch applies the event and the other skips only that event; both branches then process the same next $H = 8$ examples with identical decoding random-number states and with no additional adaptation events inside the audit window. For parameter-updating methods this intervention is the realized parameter update; for EvoLib it is the corresponding library-state modification. We define

$$V_t^{(m)} = G_{t,H}(s_t^{(m,+)}) - G_{t,H}(s_t^{(m,0)}),$$

where $s_t^{(m,+)}$ and $s_t^{(m,0)}$ are the post-event and skip-event adaptive states, respectively. Consequently,

$$\mathrm{HUR}_m = \Pr(V_t^{(m)} < 0 \mid z_t^{(m)} = 1), \qquad \mathrm{UP}_m = \Pr(V_t^{(m)} > 0 \mid z_t^{(m)} = 1),$$

and

$$\mathrm{FUU}_m = 100\,\mathbb{E}[V_t^{(m)} \mid z_t^{(m)} = 1].$$

Future-update utility is an evaluation-only quantity and is never available to the deployment-time learner or controller. For every candidate included in utility analyses, we snapshot the pre-write adaptive state w_t and replay the next $H = 8$ examples in two matched branches: one applies $\Delta_t^{(j)}$ and the other skips it. The two branches use identical future examples and decoding random-number states, and we compute

$$V_{t,\mathrm{audit}}^{(j)} = G_{t,H}(w_t + \Delta_t^{(j)}) - G_{t,H}(w_t). \tag{38}$$

These audit branches are run offline and never alter the live deployment trajectory. For FORESIGHT-TTL, candidate-level utility analyses and utility-calibration targets are computed from $V_{t,\mathrm{audit}}^{(j)}$. Method-level FUU_m, HUR_m, and UP_m used in comparisons across adaptive methods are instead computed from the realized-event audit $V_t^{(m)}$ defined above, restricted to events with $z_t^{(m)} = 1$. No additional test-time writes are applied inside either audit branch during the H-step window, so the measured difference isolates the effect of the candidate intervention at time t.

REASONSTREAM-1K is deterministic once a seed is chosen. Each of the five 200-example blocks is sampled without replacement where possible. The first two blocks use MATH test problems from algebra and geometry categories, respectively. The science block uses GPQA main questions excluding all GPQA-Diamond instances; GPQA-Diamond is reserved for endpoint evaluation. The logic block cycles through Boolean Expressions, Logical Deduction, Tracking Shuffled Objects, Dyck Languages, and Multistep Arithmetic Two from BBH. The final competition block draws from the held-out competition-mathematics pool included in the exclusion set of Appendix E. AIME 2024/2025 are evaluated independently and are never used as unlabeled adaptation examples for one another.

Every 50 deployment steps we evaluate one fixed 16-example probe batch from a held-out retention pool $\mathcal{A}_{\mathrm{ret}}$ containing 320 examples. The pool is partitioned before evaluation into 20 disjoint probe batches, so each retention example appears in at most one interleaved probe during a 1,000-step stream.

Retention probes are strictly read-only: they generate scored model responses under the current adaptive state but do not construct candidate updates, update provisional confidence sequences, contribute post-commit persistence observations, trigger promotion or rollback, or otherwise modify adaptive state. After scoring a probe, deployment resumes from exactly the pre-probe adaptive state. We additionally evaluate the full retention pool once under w_0 before deployment and once under w_{end} after the final task interaction. Both endpoint evaluations are strictly read-only. Equation (39) uses these two full-pool endpoint evaluations; the interleaved 16-example batches are used only to track retention over the course of deployment.

$\mathcal{A}_{\mathrm{ret}}$ is disjoint from $\mathcal{A}_{\mathrm{prot}}$, all controller meta-training data, and all endpoint benchmarks. We report

$$\mathrm{RET} = 100\,[\mathrm{Acc}(\mathcal{A}_{\mathrm{ret}}; w_{\mathrm{end}}) - \mathrm{Acc}(\mathcal{A}_{\mathrm{ret}}; w_0)]\,, \tag{39}$$

so negative values denote forgetting. Online-accuracy curves exclude retention probes from the x-axis.

Corruption stress-test protocol. We evaluate robustness on REASONSTREAM-1K at corruption rates $p \in \{0, 0.05, 0.10, 0.20, 0.30, 0.40\}$. Since retention probes are read-only and excluded from the learning stream, corruption is applied only to the $N = 1000$ task-learning positions. For each nonzero rate p, we select exactly $\lfloor pN \rfloor$ positions without replacement using a random permutation determined by the evaluation seed. The 5%, 10%, 20%, 30%, and 40% conditions therefore contain exactly 50, 100, 200, 300, and 400 corrupted learning positions, respectively.

At each selected position, a base corruption-operator draw and all exogenous corruption randomness are fixed by the evaluation seed before any method is run. Operators (iii) and (iv) are therefore identical across methods. Operators (i) and (ii) are applied to each method's own sampled trajectory clusters. If the drawn operator is inapplicable because fewer than two answer clusters are present for that method, we resample uniformly from the remaining applicable operators using a method-specific deterministic RNG stream derived from the same evaluation seed. Thus every selected position receives exactly one corruption, although the realized operator may differ across methods when applicability differs.

Corruption modifies only the test-time learning signal used to construct candidate updates. The original task input, the model response used for online accuracy, and the ground-truth label used

for external evaluation remain unchanged. Corruption indicators are never exposed to the learner or controller. Different evaluation seeds use independently generated corruption patterns.

G Baseline Implementations

Static. The backbone uses the same prompt and decoding settings as adaptive methods but never updates weights or memory.

Shared optimization settings. All parameter-updating baselines use rank-16 LoRA with $\alpha_{\mathrm{LoRA}} = 32$, dropout 0, and the same maximum Frobenius-norm update of 0.15 used by our adaptive candidates. Updates use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient clipping at 1.0. Learning rates are selected from the common meta-validation grid reported in Appendix I. We use constant learning rates within deployment because each online update contains only one optimization epoch.

Table 3: Optimization settings for parameter-updating baselines.

Setting	TTRL	DDRL	TTSR
Optimizer	AdamW	AdamW	AdamW
Learning rate	1×10^{-5}	3×10^{-5}	3×10^{-5}
β_1, β_2	$0.9, 0.999$	$0.9, 0.999$	$0.9, 0.999$
Weight decay	0.01	0.01	0.01
Gradient clip	1.0	1.0	1.0
LoRA rank / scale	16 / 32	16 / 32	16 / 32
Max update norm	0.15	0.15	0.15
Update buffer	16	16	32
GRPO clip	0.20	–	0.20
KL coefficient	0.005	–	0.001
Optimization epochs	1	1 + 1 refine	1

TTRL. We sample $G = 8$ responses per learning question and use majority agreement as the pseudo-reward. Every 16 deployment questions, the buffered trajectories form one GRPO update with clip ratio $\epsilon_{\mathrm{GRPO}} = 0.2$, KL coefficient $\beta_{\mathrm{KL}} = 0.005$, and one optimization epoch. The selected LoRA learning rate is 1×10^{-5}. The policy update is restricted to the shared rank-16 LoRA capacity and maximum update norm.

DDRL. We implement frequency-based ambiguity filtering and fixed-advantage debiasing following Yu et al. (2026). For an answer cluster with empirical frequency f, clusters with $f \geq 0.75$ are retained as high-consensus positives, clusters with $f \leq 0.25$ as low-frequency negatives, and intermediate-frequency clusters are discarded. Retained positive and negative trajectories receive fixed advantages $+1$ and -1, respectively, and are sampled in equal numbers whenever both are available. Every 16 deployment questions we perform one AdamW update, followed by one off-policy refinement minibatch containing up to 64 rejection-sampled trajectories from balanced positive and negative FIFO replay buffers. The selected LoRA learning rate is 3×10^{-5}, and the total number of optimized tokens is capped to the same deployment budget used for the other adaptive baselines.

TTSR. We implement a budget-matched TTSR baseline based on the v1 reflection-and-synthesis mechanism of He et al. (2026). To ensure a controlled comparison, we use the shared rank-16 LoRA capacity, meta-validation procedure, and deployment optimization budget described above, while retaining TTSR's teacher reflection and targeted-variant synthesis mechanism. Every 32 deployment questions, the model analyzes failed reasoning trajectories, summarizes recurring weaknesses, and synthesizes $M = 16$ targeted variants. For each variant, the student generates $G = 8$ trajectories. We perform one GRPO optimization epoch with clip ratio $\epsilon_{\mathrm{GRPO}} = 0.2$ and KL coefficient $\beta_{\mathrm{KL}} = 0.001$. The selected LoRA learning rate is 3×10^{-5}. Teacher synthesis uses the same temperature 0.7 and top-$p = 0.95$ generation configuration as the learning-trajectory pipeline. Student updates use the shared rank-16 LoRA capacity and maximum update norm.

EvoLib. We maintain textual skills/reflections with a 16k-token library budget. Abstraction consolidation runs every 50 interactions. Since EvoLib leaves model parameters frozen, it is particularly informative on retention.

H Task Prompts and Weak Learning Signals

For mathematical reasoning we use the system instruction "Reason carefully and return the final answer in a boxed expression." The learner samples $K = 8$ independent trajectories per learning example for pseudo-label construction. Candidate-generation decoding uses temperature 0.7, top-$p = 0.95$, no top-k filtering, and a maximum generation length of 8,192 tokens. The same candidate-generation configuration is used for all adaptive methods sharing a backbone. Sampling randomness is determined solely by the evaluation seed.

For endpoint evaluation, all methods using the same backbone share an identical decoding configuration. For DeepSeek-R1-Distill-Qwen-7B we use temperature 0.6, top-$p = 0.95$, and a maximum of 32,768 generated tokens. For Qwen3-8B we enable thinking mode and use temperature 0.6, top-$p = 0.95$, top-$k = 20$, and a maximum of 32,768 generated tokens. For Llama-3.1-8B-Instruct we use temperature 0.6, top-$p = 0.9$, and the same 32,768-token generation cap.

For AIME, we estimate pass@1 from eight stochastic responses per problem for each of four independent decoding seeds. For the remaining endpoint benchmarks, we draw one stochastic response per example for each evaluation seed and average accuracy across seeds. Decoding randomness is independent across seeds but shared across compared methods whenever possible. GPQA uses shuffled answer labels to reduce positional bias. BBH uses the canonical task wording without chain-of-thought exemplars for the streaming condition.

The self-consistency term is a sequence-level negative log-likelihood on trajectories whose final answer agrees with the modal answer, weighted by normalized vote margin. The NTP term is computed only over observed prompt and accepted trajectory tokens; no hidden ground-truth labels are used. To prevent uncontrolled language drift, the KL term in Eq. (7) is the mean token-level KL divergence over a 64-token prefix against a frozen copy of the pre-adaptation model. We use $\alpha = 0.60$ and $\lambda_{\text{kl}} = 0.02$ throughout all experiments.

I Hyperparameters

The default hyperparameter configuration used throughout the experiments is summarized in Table 4. The adaptive-baseline learning-rate selection procedure is reported separately in Table 5. We tune controller hyperparameters only on meta-validation streams generated from held-out MATH/OpenR1 problems. Benchmark test labels are used solely for final evaluation and counterfactual analysis after the run. For each adaptive baseline, the learning rate is selected from the common grid above using the same meta-validation streams and equal search budget. All other method-specific optimization settings follow Appendix G.

J Horizon Sensitivity

Update precision is computed as $\text{UP} = \mathbb{P}(V_t > 0 \mid z_t = 1)$ on the held-out evaluation streams. For this ablation only, we retrain an otherwise identical controller separately for each $H \in \{1, 2, 4, 8, 16, 32\}$; horizons are not mixed within a single controller. Figure 5 examines the horizon used to define future update utility. $H = 1$ mostly reproduces local filtering and achieves weak update precision. Utility rises sharply through $H = 8$, after which returns diminish while meta-training cost grows. This behavior motivates $H = 8$ as the default rather than claiming a universal horizon.

K Plasticity Routing Analysis

Figure 6 visualizes the fraction of accepted writes routed to each subspace. The distribution is not constant across domains. Science uses attention/output projections more frequently, logic allocates

Table 4: Default FORESIGHT-TTL hyperparameters used in the empirical study.

Quantity	Value
LoRA rank / scale	16 / 32
Candidate subspaces M	6 + no-write
Candidate learning rates $\eta_{1:6}$	$(5, 5, 8, 5, 8, 10) \times 10^{-5}$
Pseudo-label samples K	8
Weak-loss SC weight α	0.60
KL regularization λ_{kl}	0.02
Default horizon H	8 future interactions
Discount γ	0.95
LCB multiplier β	1.64
Write threshold τ_w	2.5×10^{-3} raw reward (0.25 reward points)
Interference penalty λ_i	0.30
Write-cost penalty λ_c	0.08
Sequential-CS error level α_{CS}	0.10
Commit threshold τ_{commit}	+1.5 cumulative reward units
Rollback threshold τ_{rollback}	1.0 cumulative reward units
Max provisional lifetime $K_{\max}$	32 interactions
Persistence window K_p	8 post-commit observations
Persistence discount λ_p	0.95
Promotion threshold τ_p	0
Episodic half-life	96 interactions
Persistent adapter rank budget	128 total
Controller ensemble heads	4
Endpoint temperature (DeepSeek/Qwen3/Llama)	$0.6/0.6/0.6$
Endpoint top-p (DeepSeek/Qwen3/Llama)	$0.95/0.95/0.90$
Learning-trajectory temperature	0.7
Learning-trajectory top-p	0.95
Learning-trajectory top-k	disabled
Learning-trajectory max tokens	8,192
Value Huber transition δ_V	0.10 raw utility
Interference stabilizer $\epsilon_{\cos}$	10^{-8}
Uncertainty floor ϵ_σ	10^{-4}

Table 5: Adaptive-baseline learning-rate selection.

Configuration	Value
Search grid	$\{1, 3, 10, 30, 100\} \times 10^{-6}$
TTRL selected LR	1×10^{-5}
DDRL selected LR	3×10^{-5}
TTSR selected LR	3×10^{-5}

more writes to episodic memory, and competition mathematics favors the late-block joint adapter. A fixed-subspace TTL baseline therefore pays unnecessary interference cost on domains whose useful gradient geometry differs.

L COMPONENT ABLATIONS

Figure 7 visualizes the adaptation-retention trade-off, while Table 6 reports the effect of removing individual components. The largest degradation comes from replacing future utility with current loss improvement, directly testing the paper's main hypothesis that prospective utility prediction improves update selection. Unpaired counterfactual supervision also degrades performance, consistent with the high variance of independently sampled future branches. Removing rollback leaves final endpoint

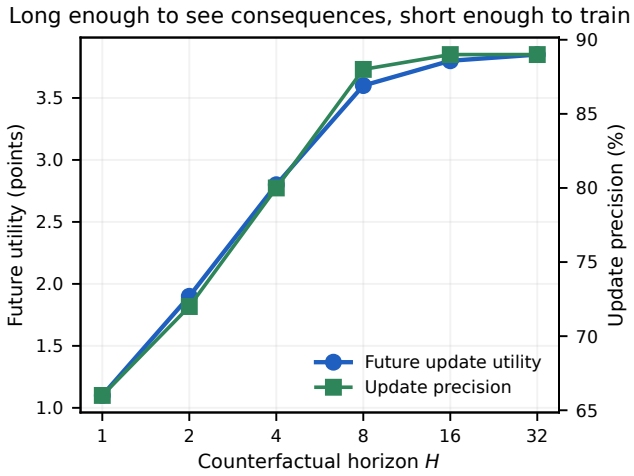

Figure 5: Sensitivity of future utility and update precision to counterfactual horizon. Longer horizons expose interference that immediate objectives miss, but eventually yield diminishing returns.

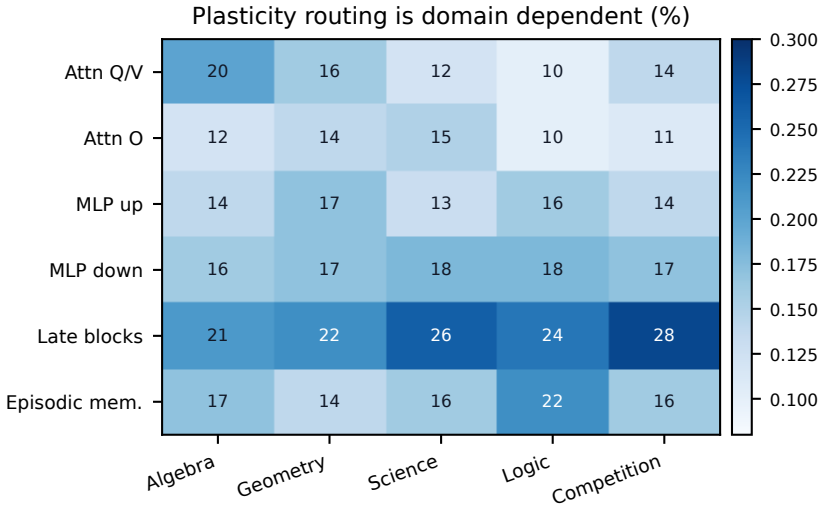

Figure 6: Accepted-write routing distribution across REASONSTREAM-1K. Values are percentages within each domain column.

accuracy relatively strong early in the stream but produces substantially worse retention after repeated switches.

M COMPUTE AND MEMORY ACCOUNTING

Measurement setup. All deployment wall-clock measurements are collected on a single NVIDIA H100 80GB GPU using BF16 inference and adaptation. All methods use the same hardware, software stack, benchmark streams, and generation-length configuration. Reported wall-clock multipliers are normalized to Static inference measured under the same configuration and include all method-specific test-time generation, backward, controller, and parameter-update costs. Offline controller meta-training is excluded.

At deployment, FORESIGHT-TTL computes the task generation, one backward pass for candidate gradients, cheap projected candidate statistics, and a small controller forward pass. We do *not* instantiate six full optimizer steps: a single base gradient is projected into candidate subspaces, and update norms/costs are computed without applying candidates. Only the selected candidate is

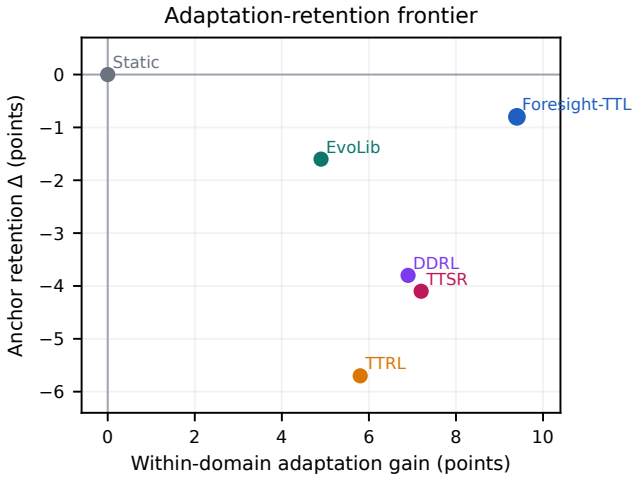

Figure 7: Adaptation-retention frontier. Better methods move right (larger within-domain gain) and up (less forgetting).

Table 6: Component ablation on REASONSTREAM-1K. Stream-AUC is mean online accuracy; negative retention means forgetting. HUR is measured under the 20% corruption stress test, while the remaining columns follow the default uncorrupted stream protocol.

Variant	Stream-AUC ↑	HUR ↓	Rollbacks / 1k	Retention ↑
Full FORESIGHT-TTL	**65.7**	**8.9**	36	**-0.8**
– future value (local gain gate)	60.8	27.4	51	-4.6
– uncertainty / LCB	63.1	18.7	68	-2.9
– subspace routing	63.8	13.9	42	-2.4
– rollback	64.2	15.1	0	-3.1
– persistence tiers	64.4	11.8	39	-2.0
Unpaired value targets	62.9	17.2	55	-2.7

materialized. On an 8B model, the controller contains 18M parameters and contributes less than 0.3% of forward-pass FLOPs measured during generation.

Table 7: Deployment resource profile on Qwen3-8B. Wall-clock is normalized to static generation under the same generation length.

Method	Wall-clock	Extra trainable state	Stored history
Static	1.00×	0	0
EvoLib	1.07×	0	16k text tokens
FORESIGHT-TTL	1.16×	18M ctrl. + rank-128 persistent + episodic/provisional LoRA deltas	64 update records (peak observed)
DDRL	1.31×	rank-16 LoRA	2 replay buffers
TTRL	1.38×	rank-16 LoRA	1 replay buffer
TTSR	1.52×	rank-16 LoRA	synthetic curricula

Paired rollouts are an offline cost: they are used during controller meta-training and, separately, for evaluation-only utility audits. Neither cost is included in the reported live-deployment wall-clock multipliers. With $M = 6$ subspaces and $H = 8$, naive branching would require 12 model states per prefix; in practice branches are run sequentially over shared KV-independent prompts and candidate deltas are low-rank, so GPU memory is bounded by two active states. This cost is comparable to meta-learning (Finn et al., 2017): substantial once, amortized over many deployments.

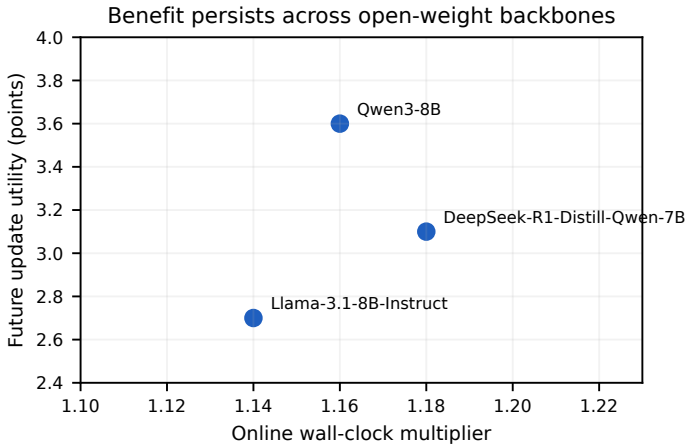

Figure 8: Future utility versus deployment wall-clock multiplier across three open-weight 7–8B backbones. The controller is retrained per backbone family but uses an identical feature interface.

N Value Calibration Protocol

After fitting Q_ψ, we calibrate only the predictive scale while leaving each ensemble mean unchanged. For head ℓ, the calibration temperature is fitted on the meta-validation set by

$$T_\ell^* = \arg\min_{T>0} \sum_{(t,j)\in\mathcal{D}_{\text{cal}}} \left[\frac{(\hat{V}_{t,\text{target}}^{(j)} - \mu_{t,\ell}^{(j)})^2}{2(T\sigma_{t,\ell}^{(j)})^2} + \log(T\sigma_{t,\ell}^{(j)}) \right]. \tag{40}$$

At deployment we replace $\sigma_{t,\ell}^{(j)}$ by $\tilde{\sigma}_{t,\ell}^{(j)} = T_\ell^* \sigma_{t,\ell}^{(j)}$ before applying Eq. (15). The calibrated temperatures are frozen before any benchmark evaluation. For a stronger safety variant, we wrap controller intervals with split conformal calibration on meta-validation residuals. The resulting interval is wider but distribution-free under exchangeability. Since deployment streams are non-exchangeable, this is not a formal guarantee for arbitrary drift; the main paper therefore treats calibration as an empirical property and uses sequential rollback as a second line of defense.

O Statistical Analysis

We use five paired evaluation seeds for stream order/decoding where the dataset size permits. For prequential endpoint benchmarks, the unit of resampling is an entire evaluation run rather than an individual example: we perform 10,000 paired bootstrap resamples over evaluation seeds and recompute the mean endpoint accuracy, keeping compared methods paired by seed.

For AIME, each evaluation run contains 30 problems with eight stochastic responses per problem. Run-level pass@1 is the mean response-level correctness over all 240 responses in the fixed prequential order. AIME uses four independent decoding seeds, and uncertainty is estimated with the same paired run-level bootstrap over seeds; problems are not resampled within a run because adaptive state couples later problems to earlier ones. Stream-AUC comparisons use a paired block bootstrap over 50-step windows to preserve temporal dependence. Harmful-update rate and update precision use the same paired block bootstrap, retaining all accepted-write outcomes within each resampled window. Blocks are resampled jointly across compared methods within each evaluation seed. Calibration curves use equal-count bins with bootstrap confidence bands.

Our primary comparisons are FORESIGHT-TTL vs DDRL for (i) Stream-AUC, (ii) harmful-update rate at 20% corruption, and (iii) held-out retention after the full stream. Endpoint benchmark accuracies are secondary. This ordering prevents selectively emphasizing whichever static benchmark happens to show the largest gain.

P Failure Modes and Falsification Tests

A prospective controller can fail in at least five ways. **Value extrapolation:** a new domain produces update features outside meta-training support, making Q_ψ overconfident. **Horizon myopia:** an update helps only after more than H steps. **Update interaction:** two individually positive updates interfere when combined. **Reference bias:** the anchor sketch over-penalizes useful specialization. **Delayed rollback:** a harmful provisional write acts for too long before enough evidence accumulates.

We therefore pre-specify five falsification tests: (1) evaluate on an unseen stream ordering and an unseen backbone; (2) sweep H through 32; (3) measure pairwise interaction error for consecutive accepted updates; (4) rerun with disjoint anchor categories; and (5) cap provisional lifetime at 4, 8, 16, and 32 interactions. If the full method does not beat a local-gain gate under these tests, the claimed advantage of prospective credit is not supported.

Q Additional Benchmark Protocols

ARC-AGI-2. Each task provides demonstration input-output grids. Each ARC grid is serialized as a rectangular JSON-style array of integer rows, e.g., `[[0,1,0],[2,2,0],[0,0,3]]`. For each demonstration pair, the prompt contains `Input:` `<grid>` followed by `Output:` `<grid>`. After all demonstration pairs, each held-out test input is rendered as `Input:` `<grid>` followed by `Output:`, which the model completes. Demonstration order follows the order provided by the benchmark and is identical across all methods. At evaluation time, we parse the first syntactically valid rectangular integer array appearing after the final `Output:` marker. Grid entries must be integers in $\{0, \ldots, 9\}$ and all rows must have equal length. Text outside the extracted array is ignored. A completion is scored incorrect if no valid array can be parsed, if the array is non-rectangular, contains out-of-range entries, or has dimensions that do not match the reference grid. No output repair, candidate selection, or parsing decision uses the reference cell values. The same serialization, prompt template, and parser are used for every compared method. For pass@1 evaluation, exactly one completion is sampled for each held-out test input under the decoding configuration specified above, and a prediction is counted correct only if the parsed grid exactly matches the corresponding reference output grid cell by cell; malformed, non-rectangular, or unparsable outputs are scored incorrect. No repair using the reference output is performed. Following the spirit of few-shot TTT (Akyürek et al., 2025), temporary candidate updates are derived only from demonstration pairs. The public evaluation grid is never used as a training label. We report single-attempt exact-grid-match accuracy (pass@1), rather than the official pass@2 metric, and use the same pass@1 protocol for every method in Table 1. FORESIGHT-TTL predicts whether the task-specific adapter should be applied to each held-out test input and resets episodic task state between unrelated ARC tasks. Persistent cross-task state is disabled in the main ARC result to avoid accidental leakage across evaluation instances. No hyperparameter, controller checkpoint, or persistent state is selected using ARC-AGI-2 evaluation tasks.

MATH-500. We use the 500-problem evaluation subset introduced by Lightman et al. (2024). Model outputs are parsed by first extracting the last valid `\boxed{}` expression, consistent with the required response format. If no valid boxed expression is present, we fall back to the final parseable mathematical expression in the completion.

Correctness is determined using Math-Verify, which normalizes LaTeX expressions, converts extracted answers to a common symbolic representation when possible, and checks mathematical equivalence against the reference answer. The same extraction and verification pipeline is used for every compared method, backbone, and evaluation seed.

AIME and competition mathematics. For AIME, final answers are integers 000–999 and exact match is used. AIME 2024 and 2025 are cited via the public Math-AI releases (Zhang & Math-AI Team, 2024; 2025). The held-out competition-mathematics pool used in REASONSTREAM-1K is constructed from public MathArena problem releases published on or before August 31, 2026 (Balunovic et al., 2025; Dekoninck et al., 2026); releases after this cutoff are excluded from all experiments.

GPQA-Diamond. Multiple-choice labels are randomly permuted per example. The learner sees only the question and sampled trajectories; ground-truth answer is unavailable to the update rule. Science-domain adaptation is especially useful for testing whether gradients from specialized knowledge corrupt general reasoning anchors.

BBH. We use the official 23-task suite but designate five logic-heavy tasks for the non-IID stream. The remaining tasks form a held-out transfer set. The BBH entry in Table 1 averages the full 23-task suite under the endpoint protocol above; the five-task/remaining-task split is used only for the REASONSTREAM-1K transfer analysis. This prevents the controller from receiving unlabeled exposure to every BBH task before transfer evaluation.