WHEN TO PROMPT, WHEN TO UPDATE: CONTEXT-PARAMETER ALLOCATION FOR LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents must follow structured protocols, adapt to evolving environments, and operate under memory, latency and cost constraints. Given these constraints, adaptation then requires deciding where task-relevant information should live; in the prompt, which is fast and reversible but bounded, or in the parameters, which is persistent but expensive to write. We formalize this as an information-allocation problem. We use prompt length as an indicator of task implementation feasibility as parameter writes have diminishing returns, while the usable prompt is bounded by the context window, for which we derive and empirically validate a proxy scaling law that provides a measurable prompt-side constraint. We propose an oracle-student-controller architecture that acts on this analysis. A compact student is first distilled to acquire the desired output schema. During deployment a controller preserves the schema by construction, projects the semantic state into a feasible prompt domain with an approximation guarantee, and triggers oracle-supervised parameter updates only on persistent error. The trigger is computable from student and environment observable signals alone, so the oracle is never queried in order to decide whether to use it. Across Multi-Fidelity Bayesian Optimization, `WebArena` tasks, and `ALFWorld`, we studied *Llama-3.1-8B* and *Mistral-7B* performance as students. Experimental results show improvement of their baseline performance by integrating the adaptive allocation schema, also compared against current state-of-the-art methods.

1 INTRODUCTION

Adapting a *Large Language Model* (LLM) to a target task is a central requirement for reliable deployment. This requirement is especially pronounced in *Agentic AI*, where LLMs interact with external tools, environments, and other agents to perform coordinated reasoning and decision-making. In these systems, the model must follows communication protocols, selects appropriate actions, and adjusts its behavior as the environment evolves. Reliability therefore depends not only on the capabilities of the underlying model, but also on mechanisms that regulate communication, detect failures, and adapt behavior over repeated interactions Ren et al. (2025).

The most immediate form of adaptation is *in-context learning* (ICL), which modifies model behavior by placing instructions, demonstrations, retrieved evidence, or interaction history in the prompt leaving model parameters fixed; prompt engineering Sahoo et al. (2024), retrieval-augmented generation (`RAG`) Lewis et al. (2020), and chain-of-thought prompting Wei et al. (2022) all exploit this capability. ICL is fast, reversible, and does not require a training pipeline, but it is brittle. Beyond the finite context windows and its attention, memory and latency costs, additional context is not semantically neutral. As the prompt grows, relevant information must compete with an increasing number of tokens. This not only can produce reaching computational limits, but also can dilute task-relevant attention and can reduce protocol validity or task performance. We refer to the resulting departure from the regime in which the deployed model uses its context reliably as *prompt-domain drift*. Compact models are particularly exposed to this effect because their effective prompt domain may be substantially smaller than their advertised context window.

A natural alternative is to move task information into the model parameters. Fine-tuning can internalize task behavior, and distillation transfers capabilities from a larger model to a compact student Xu et al. (2024). This introduces a different set of limitations. It requires suitable training signals, incurs data-collection and optimization costs, and cannot generally be performed at every interaction step; repeated updates on scarce, local, or non-stationary experience can produce catastrophic forgetting McCloskey and Cohen (1989), overwriting previously useful behavior while fitting the current task state. Distillation also commonly emphasizes static output imitation rather than the interaction skills required by agentic systems Liu et al. (2025). Consequently, placing all adaptation in the parameters is no more satisfactory than placing all adaptation in an ever-growing prompt.

Task adaptation is therefore an information-allocation problem. Transient, episode-specific information is naturally supplied through ICL, whereas stable task regularities and persistent behavioral corrections may justify parameter

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.

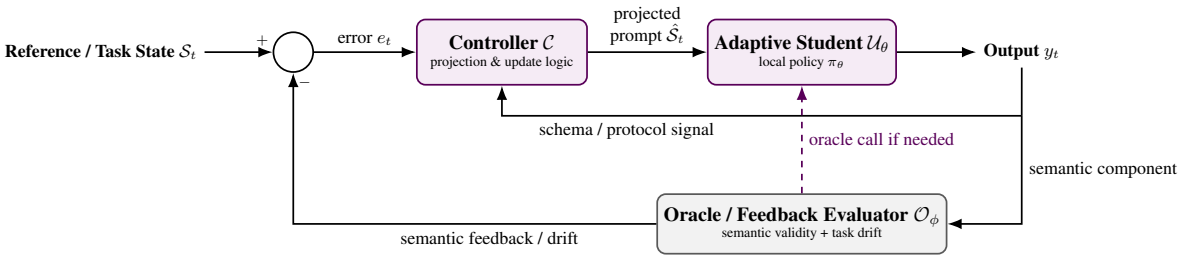


Figure 1: Control-theoretic view of the proposed framework during deployment. The adaptive student acts in the forward path and produces the output y_t. This response is decomposed into a schema/protocol component, sent directly to the controller, and a semantic component, evaluated by the oracle in the feedback path. The oracle returns a semantic feedback or drift signal, while the controller regulates prompt projection and determines when corrective intervention is required.

updates. Existing literature generally studies how to distribute information between these two paradigms based on *information quality*. Here, we approach the problem from a different perspective. In real-world systems, the main limitations often arise from system constraints. Memory capacity and latency, for instance, affect not only the quality of model outputs, but also the more fundamental question of whether the model can feasibly process the available information and produce a valid response. This is particularly important in agentic systems, where agents must not only preserve *semantic* content but also respect a communication *schema*. Thus, what matters is not only which information is retained, but also how that information is represented and delivered. Since the underlying task information cannot be measured directly, we characterize its allocation through observable dimensional proxies: prompt length for short-term contextual information and effective parameter-space dimensionality for long-term adaptation. This work therefore asks: *Can these dimensional proxies be used to dynamically decide where information should be allocated so that the model remains structurally valid throughout resource-constrained deployment?*

We answer with a framework of an oracle, a compact adaptive student, and a controller. The student is distilled offline to acquire the output schema the surrounding system requires. At deployment, the controller monitors protocol validity, self-consistency, and prompt-domain drift, projects accumulated history into a feasible prompt domain, and determines when contextual adaptation no longer suffices; only then does it trigger a lightweight, oracle-supervised parameter update. The loop treats adaptation as a controlled allocation between context and parameters: transient state is compressed in the prompt, persistent error is consolidated by selective learning (Figure 1).

We evaluate the proposed architecture in three environments: Multi-Fidelity Bayesian Optimization, where the optimal decision is computable at each step; `WebArena` Zhou et al. (2023), where the model interacts with web environments to complete tasks; and `ALFWorld` Shridhar et al. (2021), where the model acts in an embodied household environment. We compare the full architecture against its baselines, the projection mechanism against context-management methods such as MemGPT Packer et al. (2023), and the oracle-supervised adaptation against PivotRL Yi et al. (2026).

Our contributions are as follows:

- **(C1)** We formulate resource-constrained agentic adaptation as an information-allocation problem and derive an empirically computable prompt-side scaling principle based solely in information dimensionality.
- **(C2)** We give a projection operator that takes dimensionality as a *constraint* rather than a length target, preserving the output schema by construction and reducing semantic state under a knapsack budget with a $(1 - e^{-1/2})$ guarantee.
- **(C3)** We quantify an empirically computable proxy rule on catastrophic forgetting risk based on system dimensionality which serves as a trigger mechanism to allow parameters updates.
- **(C4)** We validate on MFBO, `WebArena` and `ALFWorld` and report the cost–quality trade *per environment*.

The remainder of the paper is organized as follows. Section 2 situates the framework relative to prior work. Section 3 formalizes the agentic prompt-state setting and establishes the information-allocation gap. Section 4 presents the oracle–student–controller architecture together with the projection operator and the update trigger. Section 5 reports the empirical study on MFBO, `WebArena`, and `ALFWorld`, and Section 6 concludes the paper.

2 Related Work

Four bodies of work address different parts of this problem. *Prompt-compression* methods shorten the input to a token budget via perplexity (Jiang et al., 2023), learned importance scores (Zhao et al., 2025; Chen et al., 2025a), or environment-specific trimming (Kerboua et al., 2025). *Agent-memory* methods instead decide what to retain across

interactions (Packer et al., 2023; Li et al., 2026; Zhang et al., 2026), with relevance on POES that selects evaluation examples via a facility-location objective (Ma et al., 2026). *Architectural* methods change how the model handles context from within (Munkhdalai et al., 2024; Ye et al., 2026; Geng et al., 2023; Willard and Louf, 2023), reducing the burden on our projection without deciding where information should live, while *parameter-side control* determines how a correction changes the weights (Kirkpatrick et al., 2017; Farajtabar et al., 2019; Wang et al., 2023; Cao and Wu, 2025; Ren et al., 2024; Xiong and Xie, 2026; Wang et al., 2021; Hoy and Celik, 2025) and agentic post-training selects which states carry useful signal (Yi et al., 2026; Chen et al., 2025b).

3 Problem Statement

Consider an LLM defined as $f(\rho, \theta)$, see Appendix A.1, and a task $\mathcal{A}_t$ comprising prompts $\rho_i \in \mathcal{A}_t$. We seek to maximize the model's expected task reward:

$$\max_{\theta, g(\cdot)} \mathbb{E}_{\rho \sim \mathcal{A}_t} \left[\mathrm{R} \left(f(g(\rho), \theta) \right) \right], \tag{1}$$

where $g(\cdot)$ denotes an ICL mechanism, as discussed in Section 1. As shown in Appendix A.3, optimizing equation 1 involves two complementary choices: adapting the model parameters θ or optimizing the ICL mechanism $g(\cdot)$. Under the memory interpretation of Behrouz et al. (2025), these choices correspond to placing task-relevant information $\mathcal{I}$ in long-term memory $\mathcal{M}_{\mathrm{LT}}$, encoded by θ, or short-term memory $\mathcal{M}_{\mathrm{SL}}$, supplied through $g(\cdot)$. This induces an allocation problem where memory allocation is restricted by system time and space constraints.

In order to address it, first let us define an Agentic AI system as a directed interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ is either an LLM-based agent or an external tool with local policy f_v, and each edge $(i, j) \in \mathcal{E}$ is an information channel. Graph connectivity is dynamic and modeled through $w_{i,j}(t) \in \{0, 1\}$ indicating if the channel is active 1 or inactive 0. These connectivity is governed by an external protocol system such as AG-UI Pajo (2025), A2A Habler et al. (2025) or MCP Hou et al. (2025). A relevant requirement on those protocols is the presence of an *Output Schema* of each node in order to ensure proper internal communication on $\mathcal{G}$. This allows us to introduce the following definition, Definition 1.

Definition 1 (Task-Relevant Information)**.** *For a task instance with prompt ρ, define its task-relevant information as*

$$\mathcal{I}(\rho) = \big(\rho, \Sigma_{\mathrm{out}}\big) \cup \mathcal{I}_{\mathrm{aux}}, \qquad \mathcal{I}_{\mathrm{aux}} \subseteq \mathcal{C}, \mathcal{Q}, \mathcal{H},$$

where Σ_{out} specifies the desired output schema, and $\mathcal{C}$, $\mathcal{Q}$, and $\mathcal{H}$ denote auxiliary context, examples, and interaction history, respectively. An auxiliary element is relevant if its inclusion improves the expected task reward.

From Definition 1, both ICL and fine-tuning face system limitations in injecting task-relevant information into the model. The amount of information needed for fine-tuning θ is limited by catastrophic forgetting risk, Appendix A.3, while upper-bounded by the information-transfer nonlinear behavior, Assumption 1. ICL instead injects information by expanding the input prompt and is limited by system capacity, generation may fail when the prompt exceeds the range supported by the KV cache, see Lemma 1. Additionally, performance may degrade well before this limit because long prompts can induce *model-internal saturation*. In this regime, the prompt remains technically admissible, but the task-relevant subsequence no longer exerts sufficient influence on the output, see Theorem 1.

Assumption 1 (Diminishing returns of parameter writes Kaplan et al. (2020))**.** *For a fixed model, the task-relevant information a supervised update transfers into θ grows sub-linearly in the number of supervision samples, so the marginal cost of writing a further unit of information into the parameters is non-decreasing.*

Lemma 1 (Prompt Length)**.** *For a sequence of length $|\rho|$, batch size B, L layers, and n_{kv} heads of dimension d_h with precision b_{kv}, and given available device memory $M_{\max}$, model weights M_w, and runtime overhead M_{misc}, a feasible prompt length $|\rho|_{\mathrm{feasible}}$ is upper-bounded by*

$$|\rho|_{\mathrm{feasible}} \leq \min \left\{ S_{\mathrm{model}}, \left\lfloor \frac{M_{\max} - M_w - M_{\mathrm{misc}}}{B\, L\, (2\, n_{\mathrm{kv}}\, d_h)\, b_{\mathrm{kv}}} \right\rfloor \right\}. \tag{2}$$

Proof. See Appendix A.4. □

Theorem 1 (Attention dilution at rate $\Theta(m/\ell)$)**.** *Let $\mathsf{S}_{1:\ell} = (s_1, \ldots, s_\ell) \in \mathcal{I}$ be a feasible prompt sequence induced by the greedoid $(\mathbb{V}, \mathcal{I})$, and let $t = t(\ell)$ be a query position satisfying $c\ell \leq t(\ell) \leq \ell$ for a fixed $c \in (0, 1]$. Let $\mathsf{R} \subseteq \mathsf{S}_{1:t(\ell)}$ be a fixed task-relevant subsequence with $|\mathsf{R}| = m$. If the attention scores of a head satisfy $\sigma_{tj} \in [a, b]$ for all $j \leq t(\ell)$, then*

$$\frac{m\, e^{a-b}}{\ell} \;\leq\; \sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell)} \;\leq\; \frac{m\, e^{b-a}}{c\, \ell}, \tag{3}$$

so the attention mass on R *is* $\Theta(m/\ell)$ *with explicit constants, not merely bounded above. If moreover the value vectors satisfy* $\|v_i\| \le V$ *and the map* Φ *from this head's output to the logits is* L_Φ*-Lipschitz, then ablating the contribution of* R *perturbs the logits by at most*

$$\big\|\Phi(z_t) - \Phi(z_t - z_{t,\mathsf{R}})\big\| \;\le\; L_\Phi\, V\, \frac{m\, e^{b-a}}{c\,\ell}. \tag{4}$$

Proof. See Appendix A.4. □

The role of Theorem 1 is therefore not to characterize all long-context failure, but to provide a scaling principle that makes the prompt-side limit in the information-allocation problem empirically measurable. It is relevant to highlight that equation 3 bounds the *achievable concentration ratio* on R, not the attention profile itself. Indeed, with a wide score spread a single token can still hold mass close to one, so the theorem does not say attention cannot select R. Also, not that Theorem 1 is based on fixed size m, as, indeed, *Controller Block* on Section 4 retains a bounded summary rather than the raw history. Analogously, equation 4 do not claim that a model cannot re-encode R at a later layer, and it bounds sensitivity under a Lipschitz constant that serves as an statement rather than a numerical prediction. Actually, Theorem 1 is used to show that attention dilution phenomena can arise in LLMs due to prompt length increase, but it is not saying that it necessarily happens. Table 8 measures it empirically, fitting a log–log slope of -0.94 and -0.97 against the predicted -1 across the two students, and Table 42 tests the bounded-score assumption at the level of individual heads.

Together, Assumption 1, Lemma 1, and Theorem 1 expose a fundamental information-allocation gap. Fine-tuning provides persistent storage but is constrained by limited static transfer and may lag behind a dynamic environment, whereas ICL enables rapid adaptation but is bounded by KV-cache capacity and attention dilution. Defining the missing task-relevant information as

$$\mathcal{I}_{\text{miss}}(\rho) := \mathcal{I}(\rho) \setminus \big(\mathcal{M}_{\text{SL}} \cup \mathcal{M}_{\text{LT}}\big),$$

solving equation 1 requires detecting whether $\mathcal{I}_{\text{miss}}(\rho) \neq \emptyset$ and deciding which missing elements should be supplied temporarily through $\mathcal{M}_{\text{SL}}$ or consolidated into $\mathcal{M}_{\text{LT}}$ conditioned to space and latency constraints. The next section introduces an architecture that performs this detection, allocation, and transfer over the agentic interaction graph.

4 Architecture

To address the limitations identified in the previous section, we propose a three-block architecture that detects, allocates, and transfers task-relevant information. As shown in Figure 1, the *Adaptive Student Block* $\mathcal{U}_\theta$ comprises the LLM $f_\theta(\rho)$ used to solve task $\mathcal{A}_t$. The *Controller Block* $\mathcal{C}$ manages the short-term memory supplied to the student, while the *Oracle Evaluator Block* $\mathcal{O}_\phi$ manages long-term memory.

Definition 2 (Feasible prompt space)**.** *A prompt* ρ *belongs to the feasible prompt space* $\mathcal{D}$ *of a model* π_θ *if its response* $y = f_\theta(\rho)$ *satisfies the structural and semantic constraints imposed by the target system, such as producing a valid JSON object, tool call, numerical decision, or nonempty tensor. Otherwise,* $\rho \notin \mathcal{D}$*.*

Before examining each block, we emphasize a central principle of our architecture. As said in Section 1, the information required to solve a task comprises a static component $\mathcal{D}_{\text{sch}}$ and a dynamic component $\mathcal{D}_{\text{sem}}$. Because the feasible space is unknown, dynamic semantic information should remain in the short-term memory unless it contains performance-relevant information that must be realized into static knowledge provided directly by the model.

4.1 Adaptive Student Block

The *Adaptive Student Block* stands as an architectural core as it contains the LLM $f_\theta(\rho)$ used to solve task $\mathcal{A}_t$. Although the proposed orchestration is model-agnostic, more capable models generally require less intervention from the remaining blocks. Characterizing the model's capabilities is therefore essential for designing these blocks. Lemma 1 provides a metric for quantifying the hardware limitations affecting the use of $f_\theta(\rho)$. However, as exposed previously, the model may exhibit additional limitations. Building on Theorem 1, Proposition 1 serves as an empirical metric to quantify the degradation risk induced by attention saturation based on prompt length.

Proposition 1 (Attention-induced degradation ratio)**.** *Let* $|\rho|_{\text{feasible}}$ *be the hardware-feasible prompt length of* ρ *from Lemma 1. Suppose that feasibility of the prompt domain* $\mathcal{D}$ *requires at least* $\tau > 0$ *attention mass on a task-relevant subsequence* R *of size* m*. If, for every layer* ℓ *and head* h*, the query and key vectors satisfy* $\|q_t^{(\ell,h)}\| \le Q_{\ell,h},\ \|k_j^{(\ell,h)}\| \le$

$K_{\ell,h}$, *then the attention-induced degradation ratio satisfies*

$$\Delta_\rho := \frac{|\rho|_{\text{sat}}}{|\rho|_{\text{feasible}}} \leq \frac{m}{\tau|\rho|_{\text{feasible}}} \exp\left(\max_{\ell,h} \frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}\right). \tag{5}$$

Proof. See Appendix A.4. □

Proposition 1 is a *sizing rule* where the constants $Q_{\ell,h}$ and $K_{\ell,h}$ are read off a single forward pass, so they are measured rather than fitted, as seen in Section 5 and Appendix A.5. The ratio $|\rho|_{\text{feasible}}$ follows from Lemma 1 and hardware alone. Hence, system characteristics shapes short-term memory with a proxy measure that is only length dependent.

On the long-term memory, we assume that the model has limited prior knowledge and lacks task-specific specialization, as is common among small models; see Section 5. We therefore propose a two-stage process. An initial distillation phase for static knowledge, followed by parameters fine-tuning during execution controlled by *Oracle Evaluator Block*, further details are provided in Subsection 4.3.

4.2 Controller Block

The *Controller Block* represents a novel contribution of this work. Unlike current ICL methods, proposed controller $g(\cdot)$ is system-aware. Hence, under Definition 2, the initial prompt ρ is not only enhanced with extra information, see Definition 1, but projected into model feasible domain $\mathcal{D}$.

As explained in Section 1, existing context managers compress the prompt with an external model and are not system-aware, they take no feasible domain as a constraint and protect no protocol sub-sequence. In order to address these limitations, we propose a deterministic projection approach based on Proposition 1. Given a saturation threshold τ, once the degradation ratio Δ_ρ of Proposition 1 signals that the prompt risks omitting task-relevant information R, a projection operator maps the state into a token budget κ, set from the measured $|\rho|_{\text{sat}}$ (Section 5). While Δ_ρ triggers the projection, κ sizes it such as

$$\mathcal{P}_\mathcal{O} : \mathbb{V}^* \to \mathcal{D}, \qquad \widetilde{\mathsf{S}}_t = \mathcal{P}_\mathcal{O}(\mathsf{S}_t \cap \mathcal{D}_{\text{sem}}) \oplus (\mathsf{S}_t \cap \mathcal{D}_{\text{sch}}), \tag{6}$$

where $\mathsf{S}_t \in \mathbb{V}^*$ is the accumulated sequence and $\mathcal{D} \subseteq \mathbb{V}^*$ is the feasible prompt domain of ρ. Importantly, in this work we define prompt domain uniquely from a length perspective, based on Section 1 exposed limitations. This does not ensure increasing task performance due to information quality, but ensure avoiding under perfromance due to system constraints.

Two properties distinguish $\mathcal{P}_\mathcal{O}$ from generic compressors (Jiang et al., 2023; Packer et al., 2023; Zhao et al., 2025; Chen et al., 2025a). First, $\mathcal{P}_\mathcal{O}$ acts only on $\mathcal{D}_{\text{sem}}$, leaving $\mathcal{D}_{\text{sch}}$ in the prompt by construction; and, second, it is greedy-based carrying an approximation guarantee.

Based on Krause and Hübotter (2025), we frame prompt projection as a mutual-information maximization problem. In particular, let the state at time t be decomposed as $\mathsf{S}_t = Z_{\text{sem}} \oplus Z_{\text{sch}}$, then, the controller has to define a new semantic subset $A \subseteq Z_{\text{sem}}$ such that A respect length boundaries but maximizes the similarity with respect to the complete state,

$$f(A) = \sum_{g \in Z_{\text{sem}}} w_g \max_{a \in A} \text{sim}(g, a), \qquad |A| \leq \kappa - |Z_{\text{sch}}|, \tag{7}$$

where $f(\cdot)$ stands for a weighted facility-location coverage function where sim measures the similarity between two unit signatures. Note that facility-location serves as a surrogate for mutual-information between subsets.

Proposition 2 (Budgeted greedy summarization)**.** *Suppose that $f : 2^{Z_{\text{sem}}} \to \mathbb{R}_{\geq 0}$ is normalized, monotone, and submodular, and that each unit u has token cost $|u|$. Let A_{g} be obtained by iteratively selecting the feasible unit with the largest marginal gain per token under budget κ', and let $u^\star$ be the feasible singleton with the largest value. If $\mathcal{P}_\mathcal{O}$ returns the better of A_{g} and $\{u^\star\}$, then*

$$f(\mathcal{P}_\mathcal{O}(\mathsf{S}_t)) \geq \left(1 - e^{-1/2}\right) f(A^\star) \approx 0.393\, f(A^\star),$$

where $A^\star$ maximizes f. If all units have equal token cost, the constraint reduces to a cardinality constraint and standard greedy achieves the stronger factor $1 - e^{-1}$.

Proof. The budgeted result follows from Khuller et al. (1999); Lin and Bilmes (2010), and the equal-cost result follows from Nemhauser et al. (1978). Appendix A.7.4 verifies that the required properties hold for the objective in equation 7. □

Proposition 2 shows the sub-optimality guarantees under the formulated problem equation 7. The definition of a unit depends on the environment. In MFBO, a unit is one evaluation tuple (x_i, e_i, I_i, u_i); in `WebArena`, it is one accessibility-tree node or one previous action; and in `ALFWorld`, it is one (action, observation) trajectory step, see Section 5. We represent units using $d = 32$ hashed bag-of-words signatures, except in MFBO, where the signatures contain scaled numerical fields. We assign salience 2.0 to interactive nodes, 1.5 to recent or observation-changing steps, and 1.0 to all remaining units. These are lightweight design choices rather than essential components of the controller. Indeed, Table 27 shows that replacing the signatures with learned sentence embeddings improves performance by only 0.6 points, while using uniform salience reduces it by less than one point. In contrast, removing schema protection causes a 7.9–11.2-point reduction. The main benefit therefore comes from enforcing the structural constraint $Z_{\text{sch}} \subseteq \widetilde{\mathsf{S}}_t$, rather than from a particular similarity representation or salience rule. Appendix A.7.4 gives the complete environment-specific definitions. The median-interval rule previously used for MFBO approximates this projection but does not instantiate the operator itself; Table 25 provides the corresponding comparison.

4.3 Oracle Evaluator Block

Finally, the *Oracle Evaluator Block* governs the model's long-term adaptation. As discussed above, changes in the environment and repeated projection operations may gradually degrade the performance of the adaptive student. The evaluator therefore monitors the model outputs y, identifies persistent performance drift, and consolidates relevant long-horizon information into the model parameters.

To provide a flexible evaluation policy, we employ an expert LLM capable of solving the target task. Given the current state, the oracle generates a reference output $\mathsf{S}^\star$. The collected oracle–student interactions are stored in a buffer $\mathcal{B}$ and used to update the adaptive model through supervised fine-tuning (SFT) Dong et al. (2024). Specifically, the update minimizes the KL divergence between a protocol-reweighted oracle policy and the adaptive policy

$$\mathcal{L}_{\text{dis}}(\theta, \mathcal{B}) = \mathbb{E}_{(\mathsf{S}_t, \mathsf{S}_{t+1}) \sim \mathcal{B}} \left[D_{\text{KL}}(\bar{\pi}_\phi(\cdot \mid \mathsf{S}_t) \,\|\, \pi_\theta(\cdot \mid \mathsf{S}_{t+1})) \right], \tag{8}$$

where $\bar{\pi}_\phi$ assigns greater importance to tokens that define the interaction protocol:

$$\bar{\pi}_\phi(s \mid \mathsf{S}_t) = \frac{\pi_\phi(s \mid \mathsf{S}_t) w_t(\mathsf{S}_t; y_{<t} \circ s)}{\sum_{s' \in \mathbb{V}} \pi_\phi(s' \mid \mathsf{S}_t) w_t(\mathsf{S}_t; y_{<t} \circ s')}. \tag{9}$$

We distinguish between two types of errors. For an *output-schema error*, the update emphasizes protocol-defining tokens to enforce structural compatibility with downstream tools without incurring the data and computational costs of complete semantic transfer Hsieh et al. (2023). For a *semantic-performance error*, corresponding to samples in $\mathcal{D}_{\text{sem}}$, the full KL-divergence objective is applied, $\bar{\pi}_\phi = \pi_\phi$. Alternative mechanisms are beyond the scope of this work and are discussed in Appendix A.2. It is relevant to highlight that equation 8 only updates *student* weights θ, never requiring from *oracle* logits, hence, not contradicting black-box oracle admission. As extended in Appendix A.2, the oracle distribution is unavailable, but its completions provide samples from that distribution. For each sampled completion y, we minimize the teacher-forced loss $-\log \pi_\theta(y)$. Averaged over oracle samples, this is a Monte Carlo estimator of the cross-entropy term in $D_{\text{KL}}(\bar{\pi}_\phi \| \pi_\theta)$. The omitted oracle-entropy term is independent of θ, so it does not affect optimization and oracle logits remain unnecessary for training, although the numerical KL divergence itself cannot be evaluated without them.

The oracle is activated periodically according to an evaluation axioms described in Definition 3. The decision to call it must itself be cheap, that is why Definition 3 uses only quantities the student and the environment already produce avoiding constant oracle calls. Acting on the trigger carries its own risk, as mentioned in Section 1, *catastrophic forgetting*. To avoid this phenomena, we defined a proxy measure in Definition 3 which independently of any trigger is activated, it decides if the update long-term memory will be updated.

Definition 3 (Oracle-free drift trigger)**.** *At step t the controller fires if any of the following holds:* (i) $y_t \notin \mathcal{D}_{\text{sch}}$*, i.e. the response fails the schema validator;* (ii) *the student disagrees with itself, $\mathcal{U}(y_t) > \tau_u$, for an uncertainty estimator $\mathcal{U}$ computed from the student alone;* (iii) *the environment reports no state change over a window of n steps; Signals* (i) *and* (iii) *are free. For* (ii) *the default $\mathcal{U}$ is the normalized entropy over k sampled decodings, costing $k-1$ additional local forward passes; three single-pass alternatives that cost nothing are compared in Appendix A.7.5. Calibrated values (τ_u=0.45, n=5, k=4) are given in Table 22, and the trigger's precision and recall against an oracle-in-the-loop reference are reported in Table 30.*

Definition 4 (Catastrophic Forgetting proxy, Appendix A.3)**.** *For a normalized forgetting tolerance $\epsilon > 0$, define the clipped upper bound on the null-space forgetting risk as*

$$\widehat{p}_{\text{cf}}(T; \epsilon) := \frac{q_{\mathcal{D}}}{\epsilon d_{\text{L}}} \eta^2 \sum_{s=0}^{T-1} (1 - \eta\lambda)^{2s}.$$

Table 1: The two memories in MFBO. Left: what the parameters absorb; **Sch.** is schema accuracy equation 10 at 5000 offline samples, **Sem.** the fidelity accuracy attainable online at 50 oracle-supervised samples, **Cost** the wall-clock of one such update. Right: the effective prompt domain (m=50, τ=0.1, d_h=128), with Q_{eff}, K_{eff} measured from one forward pass by the protocol of Appendix A.5.1. Full sweeps in Tables 3–13 and 6–7.

Student	Sch. ↑	Sem. ↑	Cost
Llama-3.1-8B	1.00	0.42	5:51
Mistral-7B	0.25	0.08	3:16

Student	$\lvert\boldsymbol{\rho}\rvert_{\text{feas}}$	$\boldsymbol{Q_{\text{eff}} K_{\text{eff}}}$	$\lvert\boldsymbol{\rho}\rvert^{\text{prop}}_{\text{sat}}$	$\lvert\boldsymbol{\rho}\rvert^{\text{obs}}_{\text{sat}}$	$\boldsymbol{\Delta}^{\text{obs}}_{\boldsymbol{\rho}}$
Llama-3.1-8B	4.9×10^4	8.81	2.38×10^3	2160	0.044
Mistral-7B	6.1×10^4	4.11	1.03×10^3	940	0.015

At each adaptation epoch, draw $U \sim \mathrm{Unif}(0,1)$ and trigger fine-tuning if and only if $U > \widehat{p}_{\mathrm{cf}}(T;\epsilon)$. Consequently, fine-tuning is activated with probability $1-\widehat{p}_{\mathrm{cf}}(T;\epsilon)$, thereby suppressing updates whose estimated exposure to catastrophic forgetting is large. Here, $q_{\mathcal{D}}/d_{\mathrm{L}}$ is the fraction of adapter directions unconstrained by the fine-tuning data, while T, η, and λ denote the number of updates, learning rate, and regularization strength, respectively. Table 32 tests this gate empirically against EWC, OGD, O-LoRA, I-LoRA and OPLoRA at matched update count.

Algorithm 1 Controller loop

1. At step t, obtain y_t and check trigger signals of Definition 3: schema failure, stall over n steps, saturation proximity.
2. If none fired, compute $\mathcal{U}(y_t)$ from k sampled decodings and check $\mathcal{U}(y_t) > \tau_u$.
3. If no signal fired, continue without querying the oracle or touching θ.
4. If a signal fired, query the oracle for $\mathsf{S}^\star$, add $(\mathsf{S}_t, \mathsf{S}^\star)$ to $\mathcal{B}$, and classify the error (Section 4.3).
5. Once $|\mathcal{B}| \geq B_{\min}$, compute the forgetting-risk proxy $\widehat{p}_{\mathrm{cf}}(T;\epsilon)$ of Definition 4 from the current update count T.
6. Draw $U\sim\mathrm{Unif}(0,1)$; update θ by equation 8 on $\mathcal{B}$ only if $U > \widehat{p}_{\mathrm{cf}}(T;\epsilon)$, otherwise defer and keep accumulating $\mathcal{B}$.
7. Increment T on every applied update; reset counters and buffer on the schedule of Appendix A.7.5.

5 Case Study

To evaluate the effectiveness and generality of the proposed method, we consider three complementary case studies. First, we instantiate the framework in a Multi-Fidelity Bayesian Optimization (MFBO) environment, where we assess its end-to-end performance and learning capacity while empirically examining the limitations characterized in Lemma 1 and Theorem 1. We then evaluate the method in two realistic interactive environments: `WebArena`, which comprises diverse web-based tasks, and `ALFWorld`, which considers embodied household tasks. In these environments, the proposed architecture is benchmarked against PivotRL, as a representative state-of-the-art method, and single-agent direct implementation, while its projection mechanism is compared with LLMLingua and MemGPT.

We instantiate the three-agent architecture in Figure 1 using *Llama-3.1-8B* and *Mistral-7B* as adaptive models and *GPT-5* and *GPT-5-nano* as oracle models, accessed as hosted API endpoints and queried at temperature 0 for corrections; because these are closed models we do not treat their absolute numbers as the whole claim, and repeat the end-to-end comparisons with two open-weight oracles, *Qwen2.5-72B-Instruct* and *Llama-3.3-70B-Instruct* (Table 38). The allocation ordering is unchanged under both, and the fraction of oracle-only success our system retains *rises* to 0.71–0.73 on `WebArena` and 0.93–0.94 on `ALFWorld`, since a weaker oracle is easier to approximate at a small call rate. These models are integrated into `URSA`, the Universal Research and Scientific Agent Grosskopf et al. (2025). All students are updated through rank-32 LoRA adapters on attention and MLP projections, i.e., 1.1–1.2% of the parameters, on a single NVIDIA Tesla V100 16GB device. Two allocation metrics are reported throughout: *schema validity*, the fraction of responses accepted by the protocol validator,

$$\mathrm{Acc} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{y_i \in \mathcal{D}_{\mathrm{sch}}\}}, \tag{10}$$

and the *drift with respect the oracle*, defined either as the ℓ_1 norm or cosine similarity if the task is numerical or semantic. Complete protocols, sweeps, and per-task breakdowns are deferred to Appendix A.5.

5.1 MFBO: parameters and prompt capacities

MFBO is the minimal instance of the state shape of Section 3: a growing history, a decision pairing a discrete fidelity with a continuous point, and a cost budget. The agent answers `RESULT=[model,point]` defining $\mathcal{D}_{\mathrm{sch}}$, whereas the decision itself (`model,point`) plus the previous decisions and accumulated error form $\mathcal{D}_{\mathrm{sem}}$ (Appendix A.5.1).

Table 2: Results on WebArena (812 tasks) and ALFWorld (140 seen and 134 unseen tasks) using *Llama-3.1-8B* as the student. **SR** is task success rate, **Valid** is the percentage of syntactically admissible actions, **Tok.** is the mean prompt length per step, and **Hier.** is the oracle-call rate (WebArena/ALFWorld). Uncertainty is reported as the wider half-width of the Wilson 95% interval. The p-values are paired McNemar tests against our hierarchical system, with Holm correction applied separately to each benchmark; the two ALFWorld splits are pooled for testing ($n = 274$). The second block compares methods under the same oracle budget. Full results and discordant-pair counts are reported in Tables 15–19. "-" marks no oracle calls (**Hier.**) or reference models (not tested against themselves) (**p**).

		WebArena				ALFWorld		
Configuration	**Hier.** (%)	**SR** (%)	p	**Valid**	**Tok.**	**Seen** (%)	**Unseen** (%)	p
Distill only	–	11.6 ± 2.4	1.4×10^{-24}	93.7	5.9k	61.4 ± 8.3	56.7 ± 8.5	3.0×10^{-12}
+ MemGPT	–	18.3 ± 2.8	4.1×10^{-11}	92.8	2.3k	68.6 ± 8.1	63.4 ± 8.4	4.7×10^{-8}
+ DAC	–	19.6 ± 2.9	2.6×10^{-9}	94.1	2.2k	70.0 ± 8.0	65.7 ± 8.3	5.1×10^{-7}
+ $\mathcal{P}_\mathcal{O}$, no oracle	0.0	22.3 ± 3.0	1.4×10^{-5}	97.6	2.2k	74.3 ± 7.8	70.1 ± 8.2	1.1×10^{-4}
PivotRL	–	21.9 ± 3.0	4.1×10^{-5}	96.1	5.9k	76.4 ± 7.7	72.4 ± 8.1	7.2×10^{-4}
$\mathcal{P}_\mathcal{O}$ + periodic updates	6.8/5.2	24.1 ± 3.1	4.1×10^{-3}	98.1	2.2k	78.6 ± 7.5	73.9 ± 8.0	1.0×10^{-2}
PivotRL + matched oracle	6.8/5.2	23.6 ± 3.0	1.8×10^{-3}	97.2	5.9k	79.3 ± 7.5	74.6 ± 8.0	2.2×10^{-2}
Hierarchical (ours)	6.8/5.2	$\mathbf{27.3 \pm 3.2}$	–	**99.1**	2.2k	$\mathbf{84.3 \pm 6.9}$	$\mathbf{79.9 \pm 7.6}$	–
Oracle-in-the-loop trigger	100.0	28.6 ± 3.2	0.19	99.3	2.2k	86.4 ± 6.7	82.1 ± 7.4	0.31
GPT-5 oracle only	100.0	51.4 ± 3.4	–	99.6	6.1k	92.9 ± 5.5	91.0 ± 6.0	–

Information transfer among short and long-term memories behave as the predicted from previous sections, see Table 1. The schema is static and *Llama-3.1-8B* acquires it outright from a few thousand examples, whereas the decision criterion is non-stationary so it is not fully integrated into the parameters; e.g. 50 oracle-supervised samples buy 0.42 fidelity accuracy at a cost of 82 minutes per update. Parameters are thus the right place for $\mathcal{D}_{\text{sch}}$ and an expensive place for $\mathcal{D}_{\text{sem}}$, the operative content of Assumption 1. *Mistral-7B* never completes the schema transfer (0.25 at 5000 samples), so its semantic adaptation runs on an unreliable interface.

On the other hand, observing the short-term memory, both students leave their feasible domain at 1.5–4.4% of the hardware-feasible length, so the binding constraint is neither the nominal context window nor KV-cache capacity, both an order of magnitude away. Calibrating Proposition 1 on the proposed models, see Appendix A.5.1, shows that degradation is an interval, not a cutoff, with valid and invalid responses interleaving over roughly 300 tokens. This pattern is consistent with Theorem 1, as degradation is not only a length causality (Appendix A.7.6).

5.2 MFBO: END-TO-END

We now compare the full policy against fixed allocations: everything in the parameters (distillation-only), nothing in either (non-distilled), everything in the oracle (oracle-only), with budgets set from Table 1 via Proposition 1, and with MF-EI (Huang et al., 2006) and BOCA (Kandasamy et al., 2017) which are non-LLM references, see Table 14 for the full grid. Adaptive allocation dominates every fixed allocation: with *Llama-3.1-8B* it reaches distance 12.74 ± 0.62 against 124.1 for distillation-only and 435.1 for non-distilled, closing 96% of the distillation-to-oracle gap (8.04) at 3.47–4.86% of the calls and \$0.45–2.25 against \$8.25 token cost.

Table 14 shows why neither stage substitutes for the other: the collapse from 74 to 12 evaluated points without distillation is dominated by schema violations (61.7%), whereas the distilled student's residual failures are semantic and saturation-driven (5.8%). However, on the classical references, for this one-dimensional problem, MF-EI is cheaper and more accurate than any LLM configuration. This results are consistent with our claim, whose object is a compact LLM decision component in a workflow where no closed-form acquisition exists, and motivates the next subsections move to environments with no such solver.

5.3 WEBARENA

We turn to WebArena, 812 tasks over self-hosted e-commerce, forum, software-development, and content-management sites. The prompt state is now unstructured: an accessibility tree of a few thousand tokens per step, plus the action history, against an output schema of thirteen admissible browser actions (Appendix A.7.1). This changes the allocation problem in two ways. The schema is small and static, so Phase 1 transfers it cheaply, but the semantic component grows by an entire observation at every step, so the prompt crosses $|\rho|^{\text{obs}}_{\text{sat}}$ within four to six steps and the projection becomes the binding mechanism rather than an optimization.

The baselines address long-horizon failures in different ways. MemGPT manages the context through an external memory mechanism, whereas DAC compresses the prompt using both token entropy and accumulated attention and updates these scores as compression proceeds Zhao et al. (2025). PivotRL instead changes the model parameters through local on-policy updates at informative intermediate states . Our projection $\mathcal{P}_\mathcal{O}$ acts at inference time: it retains

the protocol and task-relevant state while removing accumulated observations that would otherwise push the prompt beyond the student's feasible domain. Additional compression baselines, including LLMLingua Jiang et al. (2023) and PIS Chen et al. (2025a), which combines attention-based token sampling with sentence-level selection and a learned compression rate, are reported in Table 15.

Producing valid actions is necessary but not sufficient for the long-horizon problem. Distillation raises action validity from 46.3% to 93.7%, but success remains at 11.6% as the uncompressed prompt keeps growing. Generic memory and compression methods improve success to 18.3–19.6%, while the protocol-aware projection reaches 22.3% without the oracle. Selectively allocated oracle updates raise success to 27.3% at a 6.8% call rate; calling it every step yields 28.6% ($p = 0.19$, not significant)—the controller captures most of the benefit at a fraction of the calls. The same distinction appears against PivotRL. Our system improves `WebArena` success from 21.9% to 27.3% while using a $2.7\times$ shorter prompt. Even when PivotRL receives the same oracle budget, the hierarchical allocation remains higher (27.3% versus 23.6%). Per-site results in Table 16 support this: gains are largest on `GitLab` and `CMS`, whose observations are longest and reach saturation earliest, and smallest on `Map`, where observations stay compact.

5.4 ALFWorld

Finally, `ALFWorld` evaluates the long-horizon setting through 140 seen and 134 unseen embodied household tasks across six categories. Episodes can last up to 40 steps, and the list of admissible actions changes at every turn. Unlike in `WebArena`, the action schema is therefore dynamic: $\mathcal{D}_{\text{sch}}$ must be supplied again at each step, while only the accumulated trajectory can be compressed. This makes the separation in equation 6 essential, the controller must preserve the current action schema while reducing older interaction history. Relative to distillation alone, our system improves success by 22.9 and 23.2 points on the seen and unseen splits, and outperforms PivotRL by 7.9 and 7.5 points. Reliability does not come at the expense of generalization.

The main failure of the non-distilled student is not an invalid action but a failure to remember its own progress. As illustrated in Appendix A.7.2, the agent repeatedly revisits three of nine locations until reaching the step limit. This missing information, denoted by $\mathcal{I}_{\text{miss}}$, is neither part of the action schema nor a semantic description of the current observation; it is the record of actions and locations already explored. Context management reduces the mean trajectory from 27.4 to 17.2 steps with MemGPT, while our projection further reduces solved episodes to 13.8 steps. This is only 2.6 steps longer than the GPT-5 oracle, although the hierarchical controller invokes that oracle on just 5.2% of steps. *Mistral-7B* follows the same ordering at a lower overall success level, intervening on 12.7% of steps versus 5.2% for *Llama-3.1-8B* consistent with Table 9, the smaller effective domain needs more frequent intervention.

5.5 Oracle Efficiency and Sources of Improvement

On unseen `ALFWorld` tasks, our system achieves 79.9% success against 91.0% for the GPT-5 oracle—retaining 88% of its performance at a 5.2% call rate—and on `WebArena` reaches 27.3% against the oracle's 51.4%, retaining 53% at a 6.8% call rate. A compact student still trails the frontier model on `WebArena`: the benefit of allocation is not oracle parity but a $2.4\times$ improvement in success while consulting the oracle on fewer than one in fourteen steps. Of this 15.7-point gain, projection contributes 10.7 points and oracle/parameter updates 2.5 each (unseen `ALFWorld`: 13.4, 5.3, 4.5). Budget-matched comparisons agree: hierarchical triggering exceeds periodic updates by 3.2/6.0 points (`WebArena`/`ALFWorld`) and PivotRL by 3.7/5.3 under the same supervision budget (Appendix A.7.5).

6 Conclusions

We formulated task adaptation in resource-constrained agentic deployment as an information-allocation problem between short-term memory in the prompt and long-term memory in the parameters: parameter updates give diminishing returns, while long prompts suffer attention dilution well before the KV-cache limit. The oracle–student–controller architecture makes this allocation explicit, preserving the output schema by construction, projecting the semantic state into a measured feasible domain with an approximation guarantee, and consolidating only persistent corrections into the parameters via a risk-aware trigger. Across MFBO, `WebArena`, and `ALFWorld`, adaptive allocation beat every fixed allocation, six context-management baselines, and a parameter-only method at matched budget or supervision, retaining 0.53–0.88 of GPT-5 oracle-only success while calling the oracle on only 5–7% of steps: placing information correctly matters more than repeatedly consulting a stronger model. Three limitations follow, first, a decidable output interface, degenerating to semantic-state management where $\mathcal{D}_{\text{sch}}$ is vacuous. Second, saturation constants measured per model, since only the sizing procedure transfers, not the thresholds, so a student whose schema alone exceeds $|\rho|_{\text{sat}}$ is unsuitable rather than repairable and, finally, a trigger with imperfect recall, leaving learned-representation projections and multi-agent triggers as future work.

REFERENCES

Saeed Alaei, Ali Makhdoumi, and Azarakhsh Malekian. Maximizing sequence-submodular functions and its application to online advertising. *Management Science*, 67(10):6030–6054, October 2021. ISSN 1526-5501. doi: 10.1287/mnsc.2020.3820. URL http://dx.doi.org/10.1287/mnsc.2020.3820.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures, 2025. URL https://arxiv.org/abs/2512.24695.

Kefan Cao and Shuaicheng Wu. Orthogonal low-rank adaptation in Lie groups for continual learning of large language models, 2025. URL https://arxiv.org/abs/2509.06100.

Lizhe Chen, Binjia Zhou, Yuyao Ge, Jiayi Chen, and Shiguang Ni. PIS: Linking importance sampling and attention mechanisms for efficient prompt compression, 2025a. URL https://arxiv.org/abs/2504.16574.

Yiye Chen, Harpreet Sawhney, Nicholas Gydé, Yanan Jian, Jack Saunders, Patricio Vela, and Ben Lundell. Schema-guided scene-graph reasoning based on multi-agent large language model system, 2025b. URL https://arxiv.org/abs/2502.03450.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.12. URL https://aclanthology.org/2024.acl-long.12/.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019. URL https://arxiv.org/abs/1910.07104.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Michael Grosskopf, Russell Bent, Rahul Somasundaram, Isaac Michaud, Arthur Lui, Nathan DeBardeleben, and Earl Lawrence. Ursa: The universal research and scientific agent. *arXiv preprint arXiv:2506.22653*, June 2025. URL https://arxiv.org/abs/2506.22653. Los Alamos National Laboratory, CCS-6, T-5, and HPC-DES Divisions.

Idan Habler, Ken Huang, Vineeth Sai Narajala, and Prashant Kulkarni. Building a secure agentic ai application leveraging a2a protocol, 2025. URL https://arxiv.org/abs/2504.16902.

Md Anwar Hossen, Fatema Siddika, Juan Pablo Munoz, Tanya Roosta, and Ali Jannesari. CRAFT: Forgetting-aware intervention-based adaptation for continual learning, 2026. URL https://arxiv.org/abs/2605.05732v1.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions, 2025. URL https://arxiv.org/abs/2503.23278.

William Hoy and Nurcin Celik. STABLE: Gated continual learning for large language models, 2025. URL https://arxiv.org/abs/2510.16089.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023. Version 2, 5 Jul 2023.

Deng Huang, Theodore T. Allen, William I. Notz, and R. A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 13358–13376. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.825. URL http://dx.doi.org/10.18653/v1/2023.emnlp-main.825.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1799–1808, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Imene Kerboua, Sahar Omidi Shayegan, Megh Thakkar, Xing Han Lù, Léo Boisvert, Massimo Caccia, Jérémy Espinas, Alexandre Aussem, Véronique Eglin, and Alexandre Lacoste. FocusAgent: Simple yet effective ways of trimming the large context of web agents, 2025. URL https://arxiv.org/abs/2510.03204.

Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Andreas Krause and Jonas Hübotter. Probabilistic artificial intelligence, 2025. URL https://arxiv.org/abs/2502.05244.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Hanchen Li, Runyuan He, Qizheng Zhang, Changxiu Ji, Qiuyang Mang, Xiaokun Chen, Lakshya A Agrawal, Wei-Liang Liao, Eric Yang, Alvin Cheung, James Zou, Kunle Olukotun, Ion Stoica, and Joseph E. Gonzalez. Combee: Scaling prompt learning for self-improving language model agents, 2026. URL https://arxiv.org/abs/2604.04247.

Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 912–920, 2010.

Keliang Liu, Dingkang Yang, Ziyun Qian, Weijie Yin, Yuchi Wang, Hongsheng Li, Jun Liu, Peng Zhai, Yang Liu, and Lihua Zhang. Reinforcement learning meets large language models: A survey of advancements and applications across the llm lifecycle. *arXiv preprint arXiv:2509.16679*, 2025. Manuscript submitted to ACM.

Xiaoyu Ma, Yiwen Li, Haoyue Liu, Zhichao Wang, Ye Chen, Yongxin Guo, and Xiaoying Tang. Select smarter, not more: Prompt-aware evaluation scheduling with submodular guarantees, 2026. URL https://arxiv.org/abs/2604.11328.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention, 2024. URL https://arxiv.org/abs/2404.07143.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14:265–294, 1978.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. URL https://doi.org/10.48550/arXiv.2310.08560.

Paul Pajo. Ag-ui: Enabling user-facing ai agents through a lightweight event-based protocol for backend-frontend integration, 2025. URL https://www.researchgate.net/doi/10.13140/RG.2.2.29017.22887.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of the Sixth Conference on Machine Learning and Systems (MLSys 2023), Miami Beach, FL, USA, June 4-8, 2023*, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d245e3d06dbfca2780-Paper-mlsys2023.pdf.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*, 2024.

Yinwang Ren, Yangyang Liu, Tang Ji, and Xun Xu. Ai agents and agentic ai–navigating a plethora of concepts for future manufacturing. *Journal of Manufacturing Systems*, 83:126–133, December 2025. ISSN 0278-6125. doi: 10.1016/j.jmsy.2025.08.017. URL http://dx.doi.org/10.1016/j.jmsy.2025.08.017.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications, 2024. URL https://arxiv.org/abs/2402.07927.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Brady Steele. Subspace geometry governs catastrophic forgetting in low-rank adaptation, 2026. URL https://arxiv.org/abs/2603.02224.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

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.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL https://dl.acm.org/doi/10.5555/3600270.3602070.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models, 2023. URL https://arxiv.org/abs/2307.09702.

Yifeng Xiong and Xiaohui Xie. OPLoRA: Orthogonal projection LoRA prevents catastrophic forgetting during parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024. Version 4, 21 Oct 2024.

Xi Ye, Wuwei Zhang, Fangcong Yin, Howard Yen, and Danqi Chen. DySCO: Dynamic attention-scaling decoding for long-context language models, 2026. URL https://arxiv.org/abs/2602.22175.

Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, Jian Zhang, and Venkat Srinivasan. Pivotrl: High accuracy agentic post-training at low compute cost, 2026. URL https://arxiv.org/abs/2603.21383.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.04618.

Yi Zhao, Zuchao Li, Hai Zhao, Baoyuan Qi, and Guoming Liu. DAC: A dynamic attention-aware approach for task-agnostic prompt compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://arxiv.org/abs/2507.11942.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.

A Appendix

A.1 Large Language Models Structure

A Large Language Model (LLM) is defined as a function $f(\rho, \theta) : \rho \to \mathbb{R}^{|\mathbb{V}|}$ where ρ stands for the input prompt, θ are the internal model parameters and $\mathbb{V}$ is the vocabulary space. Direct output of $f(\rho, \theta)$ is a token in the vocabulary space $\mathbb{V}$, however, if we *remove* the last layer of the model, see Figure 2, the model $f'(\{x\}_{i=1}^{t-1}, \theta)$ is analogous to a probabilistic function such that

$$y \leftarrow p(x_t|x_{<t}) = f'(\{x\}_{i=1}^{t-1}, \theta), \qquad \forall x \in \mathbb{V}.$$

For the sake of the understanding, we will indistinctly refer to $f(\rho, \theta)$ and $f'(\rho, \theta)$ indistinctly when the context allows it. In real deployment, $f(\cdot, \theta)$ is integrated in a sequential schema such that $f(\{x\}_{i=1}^{t-1}, \theta)$ is computed until either (a) we reach a maximum of tokens generated $t = T$ or (b) the model leads to a termination character $x_t =<$ eos $>$.

As shown in Figure 2, $f(\cdot, \theta)$ is computed through a sequential procedure with 3 fundamental operations: *embedding layer*, *transformer block* and *logits layer*. Each one of them it is disseminated below.

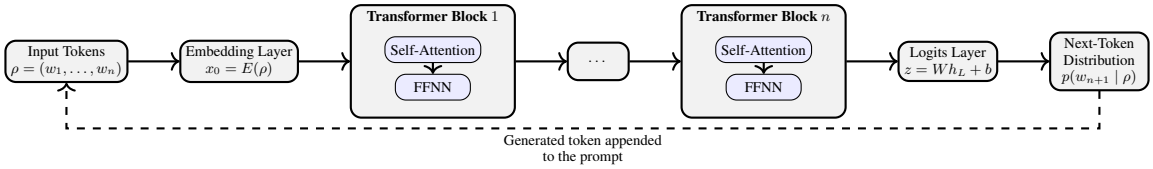


Figure 2: Schematic LLM inference procedure. Input tokens are embedded, processed through a sequence of transformer blocks containing self-attention and feed-forward neural network layers, and mapped to logits for next-token generation. The generated token is fed back into the input context for autoregressive decoding.

Embedding layer is the first and, conceptually, the most important step on the model which abstractly confers *a mapping from vocabulary space into euclidean vector space*, such that, $g_e : \rho \to \mathbb{R}^d$. Formally,

$$x^{(0)} = \mathbf{W}_e \cdot \phi(\rho) + pe(\rho) \tag{11}$$

where $\phi(\rho) \in \mathbb{R}^{|\mathbb{V}| \times n}$ tokenize the prompt, $\mathbf{W}_e \in \mathbb{R}^{d \times |\mathbb{V}|}$ embeds the tokens and $pe(\rho) \in \mathbb{R}^{d \times n}$ is the positional encoding of each token being n the number of tokens and d the embedding space.

Transformer Block is the core of any LLM. It basically operates over two distinct processes: *Attention* and *Feed Forward Neural Network*. First, **SelfAttention** is one of the most disruptive operations in the recent machine learning advances, Vaswani et al. (2017). Conceptually, attention *updates each token embedding representation by looking at the other tokens in the sequence*. Formally,

$$\hat{x}^{(\ell)} = x^{(\ell)} + \Big\{ \textbf{SelfAttention}_i\big(\mathit{Norm}(x^{(\ell)})\big) \Big\}_{i=1}^{h} \tag{12}$$

where normalization is defined as

$$\bar{x}_i^{(\ell)} = \frac{1}{\sum_{j=1 \cdots n} x_j^{(\ell)}} \cdot x_i^{(\ell)}, \qquad \forall i \in [n], \tag{13}$$

and attention is compound for three operations:

$$\textbf{SelfAttention}\big(z\big) = \mathbf{A} \cdot V \quad \in \mathbb{R}^{n \times d_h} \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the attention matrix computed as

$$\mathbf{A} = \text{Softmax}\big(\frac{Q \cdot K^\top}{\sqrt{d_h}}\big) \tag{15}$$

being $Q = \bar{x}^{(\ell)} \cdot \mathbf{W}_Q$ with $\mathbf{W}_Q \in \mathbb{R}^{d \times d_h}$ representing the *query projection* and $K = \bar{x}^{(\ell)} \cdot \mathbf{W}_K$ with $\mathbf{W}_K \in \mathbb{R}^{d \times d_h}$ representing the *key projection*. And,

$$V = \bar{x}^{(\ell)} \cdot \mathbf{W}_V \in \mathbb{R}^{n \times d_h}, \tag{16}$$

being $d_h = d/h$ where h is the amount of heads in the attention.

Then, **FFNN** is a classical feed-forward neural network operation such,

$$x^{(\ell+1)} = \hat{x}^{(\ell)} + \mathbf{FFNN}\big(\mathrm{LN}(\hat{x}^{(\ell)})\big) \tag{17}$$

with

$$\mathbf{FFNN}\big(z\big) = \mathbf{W}_2 \cdot \sigma(\mathbf{W}_1 \cdot z + \mathbf{b}_1) + \mathbf{b}_2. \tag{18}$$

Finally, given the internal representation generated after L blocks, $x^{(L)}$ must be transformed into a probability for each value in the vocabulary $\mathbb{V}$. To achieve this final mapping, we proceed on 4 operations. First,

$$x^{(L+1)} = x^{(L)} \cdot \mathbf{W}_{\text{out}} \tag{19}$$

such that $x^{(L+1)} \in \mathbb{R}^{n \times |\mathbb{V}|}$ holds, for every position in the prompt, one unnormalized score per vocabulary entry $v \in \mathbb{V}$. Next-token prediction uses the last row,

$$z \leftarrow \big(x_{n,j}^{(L+1)}\big)_{j=1}^{|\mathbb{V}|} \in \mathbb{R}^{|\mathbb{V}|}, \tag{20}$$

which are the *logits*: the index runs over the vocabulary, not over the n prompt positions. A temperature-τ softmax maps them to a distribution,

$$y_i = \frac{\mathrm{e}^{z_i/\tau}}{\sum_{j=1}^{|\mathbb{V}|} \mathrm{e}^{z_j/\tau}}, \qquad i \in \{1, \ldots, |\mathbb{V}|\}, \tag{21}$$

which is already normalized, so no further normalization is applied. Throughout, $\mathrm{LN}(\cdot)$ denotes layer normalization, used only inside the blocks above and never on a probability vector.

A.2 FINE TUNING METHODS

In this work we use Supervised Fine-Tuning **SFT** Dong et al. (2024) which imitates demonstrated actions. Whereas, there have recently arisen distinct fine-tuning baselines such Group Relative Policy Optimization **GRPO** Shao et al. (2024) which optimizes actions according to reward.

Let $\tau = (s_0, a_0^\star, \cdots, s_T, a_T^\star)$ be an interaction trajectory collected in agentic tasks, where s_i contains the prompt ρ_i and previous conversation in τ and $a_i^\star$ is the demonstrated assistant completion at that state. Then, the standard negative log-likelihood loss for **SFT** is given by

$$\mathcal{L}_{\textbf{SFT}}(\theta) = -\mathbb{E}_{\tau\sim\mathcal{D};(s_t,a_t^\star)\sim\tau}\big[\log \pi_\theta(a_t^\star|s_t)\big], \tag{22}$$

being π_θ the student agent policy. For example, if initial prompt is

$$s_0 = \text{Fix the divide-by-zero error in } \textit{calculate_average()},$$

the expected action is to inspect the code so

$$a_0^\star = \text{<tool call> } \textit{open_file("statstics.py")} \text{ </tool call>}.$$

On the other hand, let us denote π_ϕ as the reference policy. Then, we sample rollouts from π_θ and optimize the following objective such as

$$\mathcal{J}_{\textbf{GRPO}}(\theta) = \mathbb{E}_{s\sim\mathcal{D};a_i\sim\pi_\theta(\cdot|s)}\Big[\frac{1}{N}\sum_{i=1}^{N}\min\left(w_i(\theta)\hat{A}_i, \text{clip}(w_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right) - \beta\,\text{KL}\left(\pi_\theta(\cdot|s)||\pi_\phi(\cdot|s)\right)\Big] \tag{23}$$

where $w_i(\theta) = \frac{\pi'_\theta(a_i|s)}{\pi_\theta(a_i|s)}$ is the importance sampling weight between the initial and updated policy and $\hat{A}_i = \frac{r_i - \frac{1}{N}\sum_{j=1}^{N} r_j}{\text{std}\left(\{r_j\}_{j=1}^{N}\right)+\epsilon_{\text{std}}}$ represents the advantage normalized across the group of N sampled end-to-end rollouts. For example, let us suppose the same initial state than before. Instead of having a expected action, we sample multiple solutions (*trajectories*) and reward each of them such

$$\begin{aligned}
&\tau_1 \longrightarrow \text{checks for empty list, patches function, pass tests} \longrightarrow r = 3\\
&\tau_2 \longrightarrow \text{Applies the correct patch but does not run tests} \longrightarrow r = 2\\
&\tau_3 \longrightarrow \text{Catches every exception, hiding unrelated errors} \longrightarrow r = 1\\
&\tau_4 \longrightarrow \text{Changes the test instead of fixing the function} \longrightarrow r = 0,
\end{aligned}$$

hence, the update is done even if the behavior is not exact as the expected one based on the rewards.

A.3 CATASTROPHIC FORGETTING

Catastrophic forgetting is one of the main limitations that can arise when a large language model is adapted using a small fine-tuning dataset. Although closely related to overfitting, the two phenomena are not identical. Overfitting occurs when the adapted model fits the observed examples without generalizing to new examples from the same task. Catastrophic forgetting occurs when adaptation to the new task damages behavior acquired during pre-training, including behavior on prompts that do not appear in the fine-tuning dataset.

This section develops a local geometric and probabilistic characterization of this phenomenon. The central object is the null space of the Jacobian induced by the fine-tuning data after restricting the permissible parameter updates to the LoRA adaptation space. This null space contains update directions that are *invisible* (unconstrained) to the fine-tuning loss but may nevertheless modify the behavior of the pre-trained model. The resulting forgetting risk depends not only on the dimension of this null space, but also on its alignment with directions that are important for preserving the original model.

Regularized LoRA fine-tuning. As introduced in Appendix A.1, let the pre-trained LLM be represented by $f_{\theta_0}(\rho)$, where ρ is an input prompt and $\theta_0 \in \mathbb{R}^p$ denotes the pre-trained parameters. Rather than modifying all coordinates of θ_0, LoRA parameterizes selected weight matrices as

$$W = W_0 + \Delta W, \qquad \Delta W = BA,$$

where

$$A \in \mathbb{R}^{r \times d_{\text{in}}}, \qquad B \in \mathbb{R}^{d_{\text{out}} \times r}, \qquad r \ll \min\{d_{\text{in}}, d_{\text{out}}\}.$$

Let $\phi \in \mathbb{R}^{d_{\text{L}}}$ collect all trainable LoRA parameters and let

$$\theta(\phi) = \theta_0 + \mathcal{A}(\phi)$$

denote the corresponding parameterization of the adapted model, where $\mathcal{A}$ embeds the low-rank adapter updates into the complete parameter vector. The number d_{L} is the number of trainable adapter parameters and is generally much smaller than p. For example, if $W_0 \in \mathbb{R}^{100 \times 200}$ full training would imply 20.000 parameters. But using LoRA, we can define $A \in \mathbb{R}^{4 \times 200}$ and $B \in \mathbb{R}^{100 \times 4}$ and we only train 1.200 parameters.

Now, consider a fine-tuning dataset

$$\mathcal{D} = \{(\rho_i, y_i)\}_{i=1}^n.$$

For clarity, suppose that the adaptation minimizes a regularized squared-error objective

$$\mathcal{L}_{\mathcal{D},\lambda}(\phi) = \frac{1}{2n} \sum_{i=1}^{n} \left\| y_i - f_{\theta(\phi)}(\rho_i) \right\|_2^2 + \frac{\lambda}{2} \|\phi\|_2^2, \tag{24}$$

where $\lambda \geq 0$ controls the resistance to movement away from the initial adapter configuration. We take ϕ_0 to be the initial adapter parameters, typically chosen such that

$$\mathcal{A}(\phi_0) = 0 \qquad \text{and hence} \qquad \theta(\phi_0) = \theta_0.$$

More generally, the regularizer in equation 24 may be replaced by

$$\frac{\lambda}{2} \|\phi - \phi_0\|_\Gamma^2 = \frac{\lambda}{2} (\phi - \phi_0)^\top \Gamma (\phi - \phi_0),$$

where $\Gamma \succeq 0$ can encode parameter importance, for example through a Fisher-information or curvature approximation. We initially use $\Gamma = I$ and return to the general case above.

Backpropagation. Then, for each example, we can define

$$E_i(\phi) = \frac{1}{2} \left\| y_i - f_{\theta(\phi)}(\rho_i) \right\|_2^2,$$

which, by the chain rule,

$$\nabla_\phi E_i(\phi)|_{\phi=\phi_0} = -\widetilde{J}_{\phi_0}(\rho_i)^\top \left[y_i - f_{\theta_0}(\rho_i) \right], \tag{25}$$

where

$$\widetilde{J}_{\phi_0}(\rho) := \left. \frac{\partial f_{\theta(\phi)}(\rho)}{\partial \phi} \right|_{\phi=\phi_0} = J_{\theta_0}(\rho) \cdot \left. \frac{\partial \mathcal{A}(\phi)}{\partial \phi} \right|_{\phi=\phi_0} \tag{26}$$

is the effective LoRA Jacobian and maps local adapter changes into the full parameter space. Thus, LoRA does not remove the Jacobian geometry of the pre-trained model; rather, it restricts that geometry to the tangent space

$$\mathcal{S}_{\mathrm{L}} := \operatorname{range}\left(\left.\frac{\partial \mathcal{A}(\phi)}{\partial \phi}\right|_{\phi=\phi_0} \right) \subseteq \mathbb{R}^p.$$

After propagating the gradients, a first gradient-descent step with learning rate $\eta > 0$ gives

$$\begin{aligned}\Delta\phi &:= \phi_1 - \phi_0 \\ &= -\eta \, \nabla_\phi \mathcal{L}_{\mathcal{D},\lambda}(\phi)\big|_{\phi=\phi_0} \\ &= \frac{\eta}{n} \sum_{i=1}^{n} \widetilde{J}_{\phi_0}(\rho_i)^\top \left[y_i - f_{\theta_0}(\rho_i) \right] - \eta\lambda\phi_0. \end{aligned} \tag{27}$$

When $\phi_0 = 0$, the regularization term vanishes at the first step, but it acts as a restoring force at all subsequent steps. It is relevant to observe that backpropagation will only update directions present in the effective Jacobian, indeed, for a local adapter displacement $u = \phi - \phi_0$, a first-order expansion gives

$$f_{\theta(\phi_0+u)}(\rho) \approx f_{\theta_0}(\rho) + \widetilde{J}_{\phi_0}(\rho)u. \tag{28}$$

Null-space study. After defining the optimization problem and method that governs learning, we will demonstrate the importance of the null-space for catastrophic forgetting. First, let us introduce some auxiliary definitions.

Definition 5 (Fine-tuning constraint Jacobian)**.** *Let $\widetilde{J}_{\mathcal{D}}$ denote the Jacobian obtained by stacking the effective LoRA Jacobians of all fine-tuning examples:*

$$\widetilde{J}_{\mathcal{D}} = \begin{bmatrix} \widetilde{J}_{\phi_0}(\rho_1) \\ \vdots \\ \widetilde{J}_{\phi_0}(\rho_n) \end{bmatrix}.$$

If every target imposes c scalar output constraints, then $\widetilde{J}_{\mathcal{D}} \in \mathbb{R}^{nc \times d_{\mathrm{L}}}$.

Definition 6 (LoRA fine-tuning null space)**.** *The local fine-tuning null space is*

$$\mathcal{N}_{\mathcal{D}} := \ker(\widetilde{J}_{\mathcal{D}}) = \left\{ v \in \mathbb{R}^{d_{\mathrm{L}}} : \widetilde{J}_{\mathcal{D}} v = 0 \right\}.$$

Its dimension is denoted by $q_{\mathcal{D}} := \dim \mathcal{N}_{\mathcal{D}}$.

The null space contains precisely those local adapter perturbations that do not change the model outputs on the fine-tuning examples to first order, see Lemma 2.

Lemma 2 (Unconstrained LoRA directions)**.** *The dimension of the fine-tuning null space satisfies*

$$q_{\mathcal{D}} = d_{\mathrm{L}} - \operatorname{rank}(\widetilde{J}_{\mathcal{D}}) \geq d_{\mathrm{L}} - nc. \tag{29}$$

In particular, for scalar supervision, $q_{\mathcal{D}} \geq d_{\mathrm{L}} - n$. Moreover, for any $v \in \mathcal{N}_{\mathcal{D}}$ and any local update u, the updates u and $u + v$ produce identical first-order changes on the fine-tuning dataset:

$$\widetilde{J}_{\mathcal{D}}(u + v) = \widetilde{J}_{\mathcal{D}} u.$$

Proof. By the rank–nullity theorem,

$$\dim \ker(\widetilde{J}_{\mathcal{D}}) = d_{\mathrm{L}} - \operatorname{rank}(\widetilde{J}_{\mathcal{D}}).$$

Since $\operatorname{rank}(\widetilde{J}_{\mathcal{D}}) \leq nc$, the first statement follows. For the second statement, if $v \in \mathcal{N}_{\mathcal{D}}$, then $\widetilde{J}_{\mathcal{D}} v = 0$, and therefore

$$\widetilde{J}_{\mathcal{D}}(u + v) = \widetilde{J}_{\mathcal{D}} u + \widetilde{J}_{\mathcal{D}} v = \widetilde{J}_{\mathcal{D}} u.$$

□

Lemma 2 shows that replacing full fine-tuning with LoRA reduces the number of unconstrained directions from a quantity controlled by p to one controlled by d_{L}. However, LoRA does not generally eliminate the underdetermination. When $d_{\mathrm{L}} \gg nc$, many adapter directions remain invisible to the fine-tuning examples. Build upon this, let us now demonstrate the risk of forgetting.

Proposition 3 (Local possibility of forgetting)**.** *Let $v \in \mathcal{N}_{\mathcal{D}}$. For an arbitrary prompt ρ, the difference between the updates u and $u + v$ satisfies*

$$f_{\theta(\phi_0+u+v)}(\rho) - f_{\theta(\phi_0+u)}(\rho) \approx \widetilde{J}_{\phi_0}(\rho)v. \tag{30}$$

Therefore, if $\widetilde{J}_{\phi_0}(\rho)v \neq 0$ for at least one prompt outside $\mathcal{D}$, then two updates that are indistinguishable on the fine-tuning dataset can induce different behavior on the pre-trained prompt distribution.

Proof. Applying the first-order expansion equation 28 to $u + v$ and u gives

$$f_{\theta(\phi_0+u+v)}(\rho) \approx f_{\theta_0}(\rho) + \widetilde{J}_{\phi_0}(\rho)(u + v),$$
$$f_{\theta(\phi_0+u)}(\rho) \approx f_{\theta_0}(\rho) + \widetilde{J}_{\phi_0}(\rho)u.$$

Subtracting the two expressions gives equation 30 concluding the proof. □

Proposition 3 establishes the deterministic mechanism underlying the analysis: the new dataset cannot distinguish among all updates that fit its observed constraints. It does not, however, imply that every null-space direction is harmful. Some null directions may also leave the original behavior unchanged as shown in Steele (2026). Now, let us proceed to establish a measure that captures this phenomena.

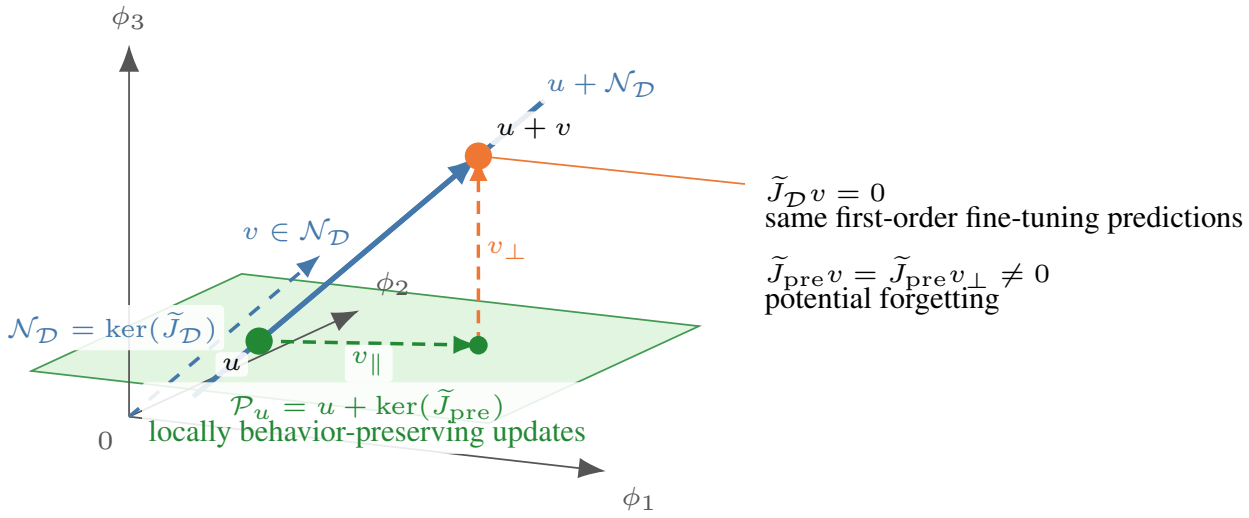


Figure 3: Local geometry of catastrophic forgetting in the LoRA adaptation space. Updates along the affine set $u+\mathcal{N}_{\mathcal{D}}$ are indistinguishable on the fine-tuning dataset to first order. Nevertheless, a null-space displacement v may contain a component $v_{\perp}$ outside the local behavior-preserving set $\mathcal{P}_u = u+\ker(\widetilde{J}_{\text{pre}})$. Consequently, $\widetilde{J}_{\mathcal{D}}v = 0$ while $\widetilde{J}_{\text{pre}}v \neq 0$, resulting in potential forgetting.

A local measure of pre-trained behavior. Let $\mathcal{P}_0$ denote a distribution of prompts representing the behavior that should be preserved. Define a preservation loss

$$\mathcal{L}_0(\phi) = \mathbb{E}_{\rho\sim\mathcal{P}_0}\left[\ell_0\left(f_{\theta(\phi)}(\rho), f_{\theta_0}(\rho)\right)\right],$$

where ℓ_0 measures deviation from the original model. The use of $f_{\theta_0}(\rho)$ as a target corresponds to behavioral preservation or functional distillation; alternatively, ground-truth labels may be used when available.

Assumption 2 (Objective Preservation)**.** *ϕ_0 is locally stationary, hence $\nabla_{\phi}\mathcal{L}_0(\phi_0) \approx 0$.*

Then, let

$$H_0 := \nabla^2_{\phi}\mathcal{L}_0(\phi_0) \succeq 0$$

denote its local Hessian, empirical Fisher matrix, or Gauss–Newton approximation in LoRA coordinates.

Definition 7 (Local catastrophic-forgetting cost)**.** *For an adapter displacement u, the local forgetting cost is*

$$\mathcal{F}(u) := \mathcal{L}_0(\phi_0 + u) - \mathcal{L}_0(\phi_0).$$

Under a second-order approximation,

$$\mathcal{F}(u) \approx \frac{1}{2} u^\top H_0 u. \tag{31}$$

Let $P_{\mathcal{N}}$ denote the orthogonal projector onto $\mathcal{N}_{\mathcal{D}}$. Every update can be decomposed as

$$u = u_{\|} + u_{\mathcal{N}}, \tag{32}$$

where

$$u_{\|} \in \operatorname{range}(\widetilde{J}_{\mathcal{D}}^\top) \quad \text{and} \quad u_{\mathcal{N}} = P_{\mathcal{N}} u \in \mathcal{N}_{\mathcal{D}}.$$

The component $u_{\|}$ is locally identifiable from the fine-tuning data, whereas $u_{\mathcal{N}}$ is not, hence, as proposed in Proposition 3, it presents a vulnerability in the training phase, potentially leading to forgetting, see Definition 8.

Definition 8 (Null-space vulnerability matrix)**.** *The old-task curvature restricted to the fine-tuning null space is*

$$H_{\mathcal{N}} := P_{\mathcal{N}} H_0 P_{\mathcal{N}}. \tag{33}$$

Its trace

$$\mathcal{V}_{\mathcal{D}} := \operatorname{tr}(P_{\mathcal{N}} H_0) = \operatorname{tr}(H_{\mathcal{N}}) \tag{34}$$

is called the null-space vulnerability.

Hence, we first defined the dimension $q_{\mathcal{D}}$ that counts how many directions the data fail to identify. Then, we specifically constrained it to $\mathcal{V}_{\mathcal{D}}$ which measures how much pre-trained sensitivity lies in those directions. Thus, nullity represents the size of the unobserved space, whereas null-space vulnerability represents its potential consequence. This distinction is relevant as, as shown in Farajtabar et al. (2019), a large null-space does not necessarily implies catastrophic forgetting, but it augments the possibilities of this space to intersect with directions of non-zero pre-trained curvature. Indeed, this distinction between safe and unsafe zones of the null-space motivated some methods such as Hossen et al. (2026) which tries to avoid catastrophic forgetting when fine-tunning.

Probabilistic null-space drift. Equation 27 describes the first full-batch update. At an arbitrary iteration t, let $\mathcal{B}_t$ denote the sampled mini batch and write its stochastic gradient as

$$\widehat{\nabla_\phi \mathcal{L}_{\mathcal{D}}}(\phi_t) = \nabla_\phi \mathcal{L}_{\mathcal{D}}(\phi_t) + \varepsilon_t,$$

where $\mathbb{E}[\varepsilon_t \mid \phi_t] = 0$, $\operatorname{Cov}(\varepsilon_t \mid \phi_t) = \Sigma_t$. Thus, the full-data gradient gives the mean update direction, while ε_t contains the perturbation introduced by the stochastic optimization procedure. Extending equation 27 to iteration t gives

$$\phi_{t+1} - \phi_t = \frac{\eta_t}{n} \sum_{i=1}^{n} \widetilde{J}_{\phi_t}(\rho_i)^\top \left[y_i - f_{\theta(\phi_t)}(\rho_i) \right] - \eta_t \lambda \phi_t - \eta_t \varepsilon_t. \tag{35}$$

Now, let $U_{\mathcal{N},t} := P_{\mathcal{N}}(\phi_t - \phi_0)$ denote the component of the accumulated adapter displacement lying in the fine-tuning null space. Consequently, projecting equation 35 onto $\mathcal{N}_{\mathcal{D}}$ and using $P_{\mathcal{N}}\phi_t = U_{\mathcal{N},t} + P_{\mathcal{N}}\phi_0$ gives

$$\begin{aligned} U_{\mathcal{N},t+1} = (1 - \eta_t \lambda) U_{\mathcal{N},t} \\ + \frac{\eta_t}{n} \sum_{i=1}^{n} P_{\mathcal{N}} \widetilde{J}_{\phi_t}(\rho_i)^\top \left[y_i - f_{\theta(\phi_t)}(\rho_i) \right] \\ - \eta_t \lambda P_{\mathcal{N}} \phi_0 - \eta_t P_{\mathcal{N}} \varepsilon_t. \end{aligned} \tag{36}$$

Equation 36 displays the sources of null-space motion directly. The factor $1 - \eta_t \lambda$ determines how much of the previous displacement is retained after regularization. The second term describes systematic leakage of the current fine-tuning gradient into the null space defined at ϕ_0. The third term appears because the regularizer in equation 27 is centered at zero rather than at ϕ_0, and the last term represents random gradient injection into unconstrained directions.

Remark 1 (Initial Configuration)**.** *Note that the initial null-space is defined over the initial configuration Jacobian ϕ_0, Definition 5, therefore, $P_{\mathcal{N}} \widetilde{J}_{\phi_0}(\rho_i)^\top = 0$.*

Remark 2 (Initially Centered Regularization)**.** *If the regularizer is centered at the initialization, $\frac{\lambda}{2}|\phi - \phi_0|^2$ then its null-space component is $\lambda U_{\mathcal{N},t}$, and no drift term involving $P_{\mathcal{N}}\phi_0$ appears.*

Under Remarks 1 and 2, equation 36 reduces locally to a frozen-Jacobian approximation such that

$$U_{\mathcal{N},t+1} = (1 - \eta_t\lambda)U_{\mathcal{N},t} - \eta_t P_{\mathcal{N}}\varepsilon_t, \tag{37}$$

where $P_{\mathcal{N}}\varepsilon_t$ should be understood as effective off-subspace perturbation caused by changing Jacobians, approximate null directions, dropout, momentum, numerical error, or other effects beyond the exact frozen-Jacobian model. Indeed, ordinary minibatch sampling alone does not generate an exact null-space component when the Jacobian is held fixed.

Assumption 3. *Let us assume a constant learning rate, zero initial drift $U_{\mathcal{N},0} = 0$ and independency on projected perturbations such $P_{\mathcal{N}}\varepsilon_t \sim \mathcal{N}\left(0, \nu^2 P_{\mathcal{N}}\right)$.*

Then, iterating equation 37, under Assumption 3, we can see that in a time horizon $T = E\left\lceil \frac{n}{B} \right\rceil$ (where E is the number of epochs and B size of batches), the displacement on the null-space is characterized as

$$U_{\mathcal{N},T} = -\eta \sum_{s=0}^{T-1} (1 - \eta\lambda)^{T-1-s} P_{\mathcal{N}}\varepsilon_s. \tag{38}$$

Consequently,

$$U_{\mathcal{N},T} \sim \mathcal{N}\left(0, \nu^2 \kappa_T P_{\mathcal{N}}\right),$$

where

$$\kappa_T := \eta^2 \sum_{s=0}^{T-1} (1 - \eta\lambda)^{2s} = \begin{cases} T\eta^2, & \lambda = 0, \\ \eta^2 \dfrac{1 - (1 - \eta\lambda)^{2T}}{1 - (1 - \eta\lambda)^2}, & \lambda > 0. \end{cases} \tag{39}$$

The quantity ν^2 measures the amount of random gradient energy injected per update, while κ_T measures how the learning rate, number of updates, and regularization determine its accumulation. After characterizing the null-space drift, we can formalize which is the amount of *forgetting* and, consequently, its probability in function of training parameters, see Theorem 2.

Theorem 2 (Forgetting under probabilistic null-space drift)**.** *Suppose that the local quadratic forgetting approximation holds and that $0 < \eta\lambda < 2$. Then*

$$\mathbb{E}[\mathcal{F}_{\mathcal{N}}(T)] = \frac{\nu^2 \kappa_T}{2} \operatorname{tr}(P_{\mathcal{N}} H_0) = \frac{\nu^2 \kappa_T}{2} \mathcal{V}_{\mathcal{D}}. \tag{40}$$

Moreover, for every forgetting tolerance $\tau > 0$,

$$\Pr\left(\mathcal{F}_{\mathcal{N}}(T) \geq \tau\right) \leq \frac{\nu^2 \kappa_T \mathcal{V}_{\mathcal{D}}}{2\tau}. \tag{41}$$

Proof. We know that,

$$\operatorname{Cov}(AX) = \mathbb{E}\big[(AX - A\mathbb{E}[X])(AX - A\mathbb{E}[X])^\top\big] = A\mathbb{E}\big[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top\big]A^\top = A \operatorname{Cov}(X) A^\top,$$

hence, as equation 38 is a linear combination of independent Gaussian perturbations, its covariance is defined as $\operatorname{Cov}(U_{\mathcal{N},T}) = \nu^2 \kappa_T P_{\mathcal{N}}$. Therefore,

$$\mathbb{E}[\mathcal{F}_{\mathcal{N}}(T)] = \frac{1}{2} \operatorname{tr}\left(H_0 \operatorname{Cov}(U_{\mathcal{N},T})\right) = \frac{\nu^2 \kappa_T}{2} \operatorname{tr}(P_{\mathcal{N}} H_0)$$

proving the first part of the theorem, while the probability bound follows from Markov's inequality. □

From Theorem 2, the local forgetting risk is governed by the amount of perturbation injected during training, the extent to which the optimization trajectory allows that perturbation to accumulate, the size of the space left unconstrained by the fine-tuning data, and the sensitivity of the original model within that space.

A training-induced forgetting index. Theorem 2 gives the task-aware forgetting measure,

$$\mathbb{E}[\mathcal{F}_{\mathcal{N}}(T)] = \frac{\nu^2 \kappa_T}{2} \mathcal{V}_{\mathcal{D}}, \qquad \Pr\left(\mathcal{F}_{\mathcal{N}}(T) \geq \tau\right) \leq \frac{\nu^2 \kappa_T \mathcal{V}_{\mathcal{D}}}{2\tau}.$$

However, computing $\mathcal{V}_{\mathcal{D}} = \mathrm{tr}(P_{\mathcal{N}} H_0)$ requires identifying how the null-space directions align with the curvature of the pre-trained behavior which requires from a sophisticated and complex control over the fine-tuning.

In order to overcome this complexity, we absorb the two task-specific quantities into a normalization rather than discarding them, so that what remains is still a bound. Since $P_{\mathcal{N}}$ is an orthogonal projector of rank $q_{\mathcal{D}}$, $\mathcal{V}_{\mathcal{D}} = \mathrm{tr}(P_{\mathcal{N}} H_0) \leq q_{\mathcal{D}}\, \lambda_{\max}(H_0)$, so $\mathbb{E}[\mathcal{F}_{\mathcal{N}}(T)] \leq \frac{1}{2}\nu^2 \kappa_T\, q_{\mathcal{D}} \lambda_{\max}(H_0)$. Defining the *normalized* forgetting $\widetilde{\mathcal{F}}_{\mathcal{N}}(T) := \dfrac{\mathcal{F}_{\mathcal{N}}(T)}{\frac{1}{2}\nu^2 d_{\mathrm{L}} \lambda_{\max}(H_0)}$ gives $\mathbb{E}[\widetilde{\mathcal{F}}_{\mathcal{N}}(T)] \leq (q_{\mathcal{D}}/d_{\mathrm{L}})\kappa_T$, which is the training-induced proxy

$$\mathfrak{E}_T := \frac{q_{\mathcal{D}}}{d_{\mathrm{L}}} \kappa_T = \frac{q_{\mathcal{D}}}{d_{\mathrm{L}}} \eta^2 \sum_{s=0}^{T-1} (1 - \eta\lambda)^{2s}$$

where $q_{\mathcal{D}}/d_{\mathrm{L}}$ is the fraction of adapter directions unconstrained by the fine-tuning data, while κ_T is the accumulated gradient-diffusion factor determined by T, η, and λ.

Therefore, for any normalized tolerance $\epsilon > 0$, Markov's inequality gives

$$\Pr\left(\widetilde{\mathcal{F}}_{\mathcal{N}}(T) \geq \epsilon\right) \leq \frac{\mathfrak{E}_T}{\epsilon}.$$

Thus, $\mathfrak{E}_T$ isolates the part of the forgetting risk controlled by the training design. If $q_{\mathcal{D}} = 0$, then $P_{\mathcal{N}} = 0$ and both $\mathfrak{E}_T$ and the corresponding null-space forgetting vanish.

Two limits should be read together with it. It bounds the *normalized* forgetting, and the normalizer $\frac{1}{2}\nu^2 d_{\mathrm{L}} \lambda_{\max}(H_0)$ is not estimated, so $\mathfrak{E}_T$ orders update policies at fixed ν^2 and H_0 rather than giving an absolute risk. And the bound is loose by construction, since $\mathrm{tr}(P_{\mathcal{N}} H_0) \leq q_{\mathcal{D}} \lambda_{\max}(H_0)$ is tight only when the null space aligns with the top eigendirection of H_0. Both are acceptable for the use made of it in Definition 4, which needs a comparable exposure score rather than a calibrated probability, and Appendix A.7.5 (Table 32) tests the gate against schedules and continual-learning baselines of *equal update count* so the comparison does not rest on the bound being tight.

A.4 PROOFS

The main text uses three notions that are standard for sequences but not for sets, collected here so that no statement rests on an undefined term. They are analytical devices making prefix-dependent feasibility well posed, which set-based or matroid constraints cannot express; no algorithm enumerates them.

Definition 9 (Greedoid over sequences)**.** *Let $\mathbb{V}$ be a vocabulary and $\mathcal{I} \subseteq \mathbb{V}^*$ a collection of finite sequences. The pair $(\mathbb{V}, \mathcal{I})$ is a* greedoid *if $\varnothing \in \mathcal{I}$ and, for every $\mathsf{S} \in \mathcal{I}$ with $|\mathsf{S}| > 0$, some prefix-deletion of S remains in $\mathcal{I}$ (accessibility), and for $\mathsf{S}, \mathsf{T} \in \mathcal{I}$ with $|\mathsf{S}| > |\mathsf{T}|$ there is $s \in \mathsf{S}$ with $\mathsf{T} \oplus s \in \mathcal{I}$ (exchange), where $\oplus$ is concatenation.*

Feasibility of a prompt depends on the order of its parts, not only which parts are present: a schema block placed after 10^4 tokens of history is not the same object as the same block placed first. A matroid cannot represent that; a greedoid can, which is the only reason the structure appears.

Definition 10 (Prefix monotonicity and string submodularity (Alaei et al., 2021))**.** *$f : \mathbb{V}^* \to \mathbb{R}_{\geq 0}$ is* normalized *if $f(\varnothing) = 0$;* prefix-monotone *if $f(\mathsf{S}) \leq f(\mathsf{S} \oplus \mathsf{T})$; and* string submodular *if for all S, T and $s \in \mathbb{V}$, $f(\mathsf{S} \oplus s) - f(\mathsf{S}) \geq f(\mathsf{S} \oplus \mathsf{T} \oplus s) - f(\mathsf{S} \oplus \mathsf{T})$.*

The selection problem solved by $\mathcal{P}_{\mathcal{O}}$ is a *set* problem, since the coverage function below does not depend on order; the sequence formalism is used only for feasibility of the resulting prompt.

Lemma 1. By Pope et al. (2023), the required cache memory is $M_{\mathrm{KV}}(|\rho|) = B\,|\rho|\,L\,(2\,n_{\mathrm{kv}}\,d_h)\,b_{\mathrm{kv}}$, which gives the stated bound. Let us suppose the following prompt $\rho =$ Summarize the following report in three concise bullet points. First, assuming a *cl100k_base* tokenizer, this is broken into 12 tokens

$$\rho' = \{\text{sum, mar, ize, the, follow, ing, report, in, three, concise, bullet, points}\}.$$

For each prompt token, every transformer layer L stores one key vector and one value vector for each of the n_{kv} heads, see Appendix A.1. Thus, the KV-cache memory needed for the prompt ρ is computed as

$$M_{kv}(|\rho|) = B \cdot |\rho| \cdot L \cdot (2 \cdot n_{kv} \cdot d_h) \cdot b_{kv}$$

where d_h is the dimension of the heads, B stands for the batch size (1 in this case) and b_{kv} are the bytes per value of each token, for example, $b_{kv} = 2$ in FP16 quantization. Now, notice that the cache total space M_{max} is not fully empty for the prompts, but it also requires to allocate model static weights M_w and the cache generated during the runtime M_{misc} concluding the example. □

Lemma 3 (Two-sided causal attention dilution)**.** *Let $\mathsf{S}_{1:\ell} = (s_1, \ldots, s_\ell) \in \mathbb{V}^*$ be a feasible prompt sequence induced by the greedoid $(\mathbb{V}, \mathcal{I})$, and consider an autoregressive self-attention head with causal masking. Let $t = t(\ell)$ satisfy $c\,\ell \leq t(\ell) \leq \ell$ for a fixed $c \in (0, 1]$, and let $\alpha_{ti}^{(\ell)}$ denote the causal self-attention weight from query position $t(\ell)$ to token s_i, $i \leq t(\ell)$. If $\sigma_{tj} = q_t^\top k_j/\sqrt{d} \in [a, b]$ for all $j \leq t(\ell)$, then for any $\mathsf{R} \subseteq \mathsf{S}_{1:t(\ell)}$ with $|\mathsf{R}| = m$,*

$$\frac{m\,e^{a-b}}{\ell} \;\leq\; \sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell)} \;\leq\; \frac{m\,e^{b-a}}{c\,\ell}.$$

Proof. Causal masking restricts attention to the prefix $\{1, \ldots, t(\ell)\}$, so the self-attention weight assigned to token s_i, with $i \leq t(\ell)$, is

$$\alpha_{ti}^{(\ell)} = \frac{\exp\left(q_t^\top k_i/\sqrt{d}\right)}{\sum_{j=1}^{t(\ell)} \exp\left(q_t^\top k_j/\sqrt{d}\right)}, \tag{42}$$

as shown in equation 15. By the bounded-score assumption $\sigma_{tj} \in [a, b]$ for all $j \leq t(\ell)$, bounding numerator and denominator in each direction gives, for every $i \leq t(\ell)$,

$$\frac{e^a}{t(\ell)\,e^b} \;\leq\; \alpha_{ti}^{(\ell)} \;\leq\; \frac{e^b}{t(\ell)\,e^a}, \qquad \text{i.e.} \qquad \frac{e^{a-b}}{t(\ell)} \leq \alpha_{ti}^{(\ell)} \leq \frac{e^{b-a}}{t(\ell)}.$$

Both sides are decreasing in $t(\ell)$, so they must be relaxed using *opposite* ends of the admissible window $[c\ell, \ell]$: substituting the largest admissible value $t(\ell) \leq \ell$ into the lower bound gives $\alpha_{ti}^{(\ell)} \geq e^{a-b}/\ell$, while substituting the

smallest admissible value $t(\ell) \geq c\ell$ into the upper bound gives $\alpha_{ti}^{(\ell)} \leq e^{b-a}/(c\ell)$; using the same substitution in both directions, as one might naively do, discards the lower bound entirely. Summing over the m elements of R,

$$\frac{m\, e^{a-b}}{\ell} \;\leq\; \sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell)} \;\leq\; \frac{m\, e^{b-a}}{c\,\ell},$$

which is equation 3. The two constants differ only through the width of the window: at $t(\ell) = \ell$, i.e. $c = 1$, both sides share the denominator ℓ. □

Theorem 1. The two-sided attention bound is Lemma 3. For the logit statement, decompose the head's attention output at position t, as in equation 14, as

$$z_t^{(\ell)} = \sum_{i=1}^{t(\ell)} \alpha_{ti}^{(\ell)} v_i = \underbrace{\sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell)} v_i}_{z_{t,\mathsf{R}}^{(\ell)}} + \underbrace{\sum_{s_i \notin \mathsf{R}} \alpha_{ti}^{(\ell)} v_i}_{z_{t,\mathsf{R}^c}^{(\ell)}}.$$

By the triangle inequality and the uniform value bound $\|v_i\| \leq V$,

$$\big\|z_{t,\mathsf{R}}^{(\ell)}\big\| \;\leq\; V \sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell)} \;\leq\; V\, \frac{m\, e^{b-a}}{c\,\ell},$$

using the upper bound of Lemma 3. Let Φ denote the composition of everything above this head, from its output z_t to the logits (as introduced in Appendix A.1), and suppose Φ is L_Φ-Lipschitz. Ablating the contribution of R replaces z_t by $z_t - z_{t,\mathsf{R}}^{(\ell)}$, so

$$\big\|\Phi(z_t) - \Phi(z_t - z_{t,\mathsf{R}}^{(\ell)})\big\| \;\leq\; L_\Phi \big\|z_{t,\mathsf{R}}^{(\ell)}\big\| \;\leq\; L_\Phi V\, \frac{m\, e^{b-a}}{c\,\ell},$$

which is equation 4. For fixed m the right-hand side is $O(1/\ell)$, and the lower bound of Lemma 3 makes the attention mass itself $\Omega(m/\ell)$, so the rate is tight in ℓ and not merely an upper estimate. □

Proposition 1. Let us denote the attention logit assigned by head h in layer ℓ from query position t to token s_j as

$$\sigma_{tj}^{(\ell,h)} = \frac{q_t^{(\ell,h)\top} k_j^{(\ell,h)}}{\sqrt{d_h}},$$

where $q_t^{(\ell,h)}$ and $k_j^{(\ell,h)}$ stand for the query and key vector respectively, while d_h is head dimension. Then, defining $Q_{\ell,h} = \sup_t ||q_t^{(\ell,h)}||$ and $K_{\ell,h} = \sup_t ||k_t^{(\ell,h)}||$, by Cauchy–Schwarz inequality,

$$\left|q_t^{(\ell,h)\top} k_j^{(\ell,h)}\right| \leq \|q_t^{(\ell,h)}\|\, \|k_j^{(\ell,h)}\| \leq Q_{\ell,h} K_{\ell,h}.$$

Hence, for each layer and head, assuming bounded-score such that $\sigma_{tj}^{(\ell,h)} \in [a_{\ell,h}, b_{\ell,h}]$ with $a_{\ell,h}, b_{\ell,h} \in \mathbb{R}$, the attention-logit spread satisfies $\big(b_{\ell,h} - a_{\ell,h}\big) \leq \frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}$.The saturation quantity of interest is the next generated token, i.e. the last prompt position, so $t(\ell) = \ell$ and the window constant of Lemma 3 is $c = 1$. Then, applying the upper bound of Lemma 3 to each layer and head gives

$$\sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell,h)} \leq \frac{m e^{b_{\ell,h} - a_{\ell,h}}}{|\rho|} \leq \frac{m}{|\rho|} \exp\left(\frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}\right),$$

which averaging over all layers and heads preserves the bound with the worst-case layer-head pair,

$$A_{\mathcal{U}}(\rho) := \frac{1}{LH} \sum_{\ell=1}^{L} \sum_{h=1}^{H} \sum_{s_i \in \mathsf{R}} \alpha_{ti}^{(\ell,h)} \leq \frac{m}{|\rho|} \exp\left(\max_{\ell,h} \frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}\right).$$

Since feasibility requires $A_{\mathcal{U}}(\rho) \geq \tau$, any feasible prompt must satisfy $\tau \leq \frac{m}{|\rho|} \exp\left(\max_{\ell,h} \frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}\right)$, such that rearranging gives

$$|\rho|_{\text{sat}} \leq \frac{m}{\tau} \exp\left(\max_{\ell,h} \frac{2Q_{\ell,h}K_{\ell,h}}{\sqrt{d_h}}\right).$$

Finally, dividing by $|\rho|_{\text{feasible}}$ the proof is concluded. □

A.5 CASE STUDY

This appendix develops the experimental evidence for Section 5 in the order needed to understand the allocation problem. We first explain the evaluation protocol and measure what the students can learn and how much context they can use. We then examine how the projection selects information, before following the system from the controlled MFBO testbed to browser interaction in `WebArena` and household tasks in `ALFWorld`. The remaining experiments ask which failures require an oracle correction or a parameter update, how much those mechanisms cost, and whether the findings persist with different oracles, decoders, and architectures. Throughout, preserving an instruction in the prompt is distinct from guaranteeing that the model follows it; the former is a property of the projection, whereas the latter is measured by the output validators.

A.5.1 MFBO: MEASURING WHAT THE STUDENTS CAN LEARN AND RETAIN

MFBO provides a controlled setting in which the interface stays fixed while the decision state grows. The agent maximizes an unknown objective $F : X \to \mathbb{R}$ using approximations $\{F_p\}_{p\in\mathcal{P}}$ with accuracy $\alpha(p)$ and execution time $t_p(\mathbf{x})$, with four fidelity choices cost 1, 2, 3, and 4 minutes.. At each iteration it returns `RESULT=[model,point]` under a wall-clock budget. As shown below, the prompt contains model information, evaluated points, errors, the active domain, uncertainty, and remaining time. The response grammar and fidelity table form the fixed interface; the other quantities describe the evolving optimization state. Individual state variables need not increase, but their accumulated history makes the prompt longer.

MFBO prompt construction

System. You got this information which is a list of values on format (i,fidelity,cpu time): model info. This indicates the model i has a certain fidelity with respect to other models and takes a certain time to find the solution. Our goal is to discover a function through sampling. Since now, the evaluated points are *eval points* with a distance to the optimal solution of *error* over the domain *domain*; the uncertainty is *uncertainty*. Select the next point to evaluate and which fidelity model should be used considering only *remaining time* is left. Strictly return the solution as `RESULT=[model,point]`.

We first isolate schema learning from decision quality. Tables 3 and 4 vary the number of Phase 1 examples and training epochs. Accuracy follows equation 10, and is itself the evidence for Assumption 1. It stays flat while the dataset is small, then rises sharply once enough supervision accumulates, the sub-linear transfer the assumption describes. Training times use hours and minutes, with 10:00 as the execution limit.

Table 3: Effect of dataset size and number of epochs on the schema distillation of *Llama-3.1-8B*.

Data size	Epochs	Accuracy ↑	Time ↓
1000	5	0.05	2:24
	10	0.10	4:48
2000	5	0.05	3:56
	10	0.15	5:30
3000	5	0.10	5:16
	10	0.20	6:40
4000	5	0.40	6:38
	10	0.60	8:15
5000	5	1.00	8:35
	10	1.00	10:00

Table 4: Effect of dataset size and number of epochs on the schema distillation of *Mistral-7B*.

Data size	Epochs	Accuracy ↑	Time ↓
1000	5	0.05	2:06
	10	0.10	4:12
2000	5	0.10	3:26
	10	0.05	4:49
3000	5	0.10	4:36
	10	0.00	5:50
4000	5	0.15	5:48
	10	0.05	7:31
5000	5	0.20	6:45
	10	0.25	8:52

The students respond differently to the same training budget. *Llama-3.1-8B* reaches full accuracy on the schema probe at 5000 examples, whereas *Mistral-7B* remains unreliable and does not improve consistently with more epochs. This establishes a limitation of the tested Mistral training configuration, rather than proving an intrinsic capacity ceiling. Its later decision results must be interpreted alongside this incomplete interface transfer.

Next, we seek to find the saturation limits by holding the distilled checkpoints fixed and increase prompt length. Table 5 distinguishes the hardware memory estimate $|\rho|_{\text{feasible}}$ from the proposed attention-based estimate $|\rho|_{\text{sat}}^{\text{prop}}$ and the observed operating threshold $|\rho|_{\text{sat}}^{\text{obs}}$. The latter two concern reliable behavior, which can deteriorate well before memory is exhausted. Tables 6 and 7 show a transition region rather than a sharp cutoff. *Llama-3.1-8B* sometimes returns a valid action after its first empty response, while *Mistral-7B* begins producing text outside the required protocol. The observed threshold therefore summarizes the onset of unreliable behavior; it does not mean that every longer prompt fails. These sweeps motivate a bounded prompt, but do not alone identify attention dilution as the cause.

Table 5: Model dimensions, memory assumptions, and prompt-length estimates for $m = 50$ and $\tau = 0.1$. The hardware estimates use the listed 24GB memory budget. Proposed and observed saturation lengths refer to the calibration sweep and are not nominal context-window limits.

Model	L	n_{kv}	d_{h}	b_{kv}	Prec.	M_{max} (GB)	M_{w} (GB)	M_{misc} (GB)	$\lvert\rho\rvert_{\text{feasible}}$	Q_{eff}	K_{eff}	$\lvert\rho\rvert_{\text{sat}}^{\text{prop}}$	$\lvert\rho\rvert_{\text{sat}}^{\text{obs}}$	$\Delta_{\rho}^{\text{obs}}$
Llama-3.1-8B	32	8	128	2	fp16	24	16	2	$4.9{\cdot}10^4$	3.25	2.71	$2.38{\cdot}10^3$	2160	0.044
Mistral-7B	32	8	128	2	fp16	24	14.5	2	$6.1{\cdot}10^4$	2.22	1.85	$1.03{\cdot}10^3$	940	0.015

Table 6: Response of distilled *Llama-3.1-8B_5000_5* as a function of prompt length. Column four is the ratio to the hardware bound ($|\rho|_{\text{feasible}} \approx 49{,}152$ tokens), column five the ratio to the Proposition 1 estimate ($|\rho|_{\text{sat}}^{\text{prop}} \approx 2.38{\cdot}10^3$). "–" indicates an empty response.

Index	**Result**	$\lvert\rho\rvert$	$\lvert\rho\rvert/\lvert\rho\rvert_{\text{feasible}}$	$\lvert\rho\rvert/\lvert\rho\rvert_{\text{sat}}^{\text{prop}}$
1	[3, 9.6323]	2003	0.041	0.842
2	[2, 3.6239]	2053	0.042	0.863
3	[3, 7.0572]	2107	0.043	0.885
4	[3, 4.8469]	2160	0.044	0.908
5	–	2211	0.045	0.929
6	[2, 2.3688]	2261	0.046	0.950
7	[1, 7.4646]	2312	0.047	0.971
8	–	2365	0.048	0.994
9	–	2416	0.049	1.015
10	–	2467	0.050	1.037

Table 7: Response of distilled *Mistral-7B_5000_10* as a function of prompt length, with $|\rho|_{\text{feasible}} \approx 61{,}440$ and $|\rho|_{\text{sat}}^{\text{prop}} \approx 1.03{\cdot}10^3$ tokens. "–" indicates an empty response and "x" a non-protocol response.

Index	**Result**	$\lvert\rho\rvert$	$\lvert\rho\rvert/\lvert\rho\rvert_{\text{feasible}}$	$\lvert\rho\rvert/\lvert\rho\rvert_{\text{sat}}^{\text{prop}}$
1	[1, 42.1200]	867	0.014	0.842
2	[3, 35.1600]	940	0.015	0.913
3	x	987	0.016	0.958
4	x	1040	0.017	1.010
5	x	1093	0.018	1.061
6	x	1147	0.019	1.114
7	x	1216	0.020	1.181
8	x	1267	0.021	1.230
9	x	1320	0.021	1.282
10	x	1344	0.022	1.305

In order to examine this phenomena carefully, we designed an experiment by keeping the relevant block fixed while adding irrelevant information extending prompt length, consistent with dilution proportional to m/ℓ over the tested range in equation 3. As seen in Table 8, the broad theoretical envelope expresses a worst-case bound rather than a precise performance prediction. However, we can see high correspondence with the predicted -1 rate in the behavior, implying a direct relation between prompt growth and attention-dilution.

Table 8: Attention mass on a fixed $m = 31$ token task-relevant block at the final prompt position, averaged across layers and heads. Irrelevant filler increases the total length. The envelope comes from equation 3; the fitted log–log slope is compared with the predicted value -1.

$\lvert\rho\rvert$	mass on R		Envelope	
	Llama-3.1-8B	Mistral-7B	lower	upper
256	0.1174	0.1038	0.0091	0.821
384	0.0806	0.0699	0.0061	0.547
512	0.0623	0.0538	0.0046	0.410
768	0.0428	0.0362	0.0030	0.274
1024	0.0331	0.0274	0.0023	0.205
1536	0.0229	0.0186	0.0015	0.137
2048	0.0176	0.0141	0.0011	0.103
3072	0.0122	0.0096	0.0008	0.068
fitted slope	-0.94	-0.97	predicted -1	
95% CI	$[-1.04, -0.84]$	$[-1.09, -0.85]$		

Then, we test whether the sizing procedure in Proposition 1 gives useful estimates beyond the two main students. For each model, a forward pass on a 64-unit prompt measures query and key norms after their linear projections. The calibration takes maxima over positions and uses the worst layer–head pair, together with the configured head dimension d_h. Table 9 fits the remaining scalar on five models and predicts the sixth, so each prediction is evaluated on a model excluded from that fit.

Table 9: Leave-one-model-out calibration across six students. The scalar is fitted on the other five models. Signed error compares predicted and observed saturation length; the final row summarizes absolute errors.

Model	$Q_{\mathrm{eff}}K_{\mathrm{eff}}$	d_h	$\lvert\rho\rvert^{\mathrm{pred}}_{\mathrm{sat}}$	$\lvert\rho\rvert^{\mathrm{obs}}_{\mathrm{sat}}$	Error
Qwen2.5-0.5B-Instruct	2.94	64	402	470	-14.5%
Llama-3.2-1B-Instruct	3.38	64	681	610	$+11.6\%$
Llama-3.2-3B-Instruct	4.02	128	1104	1290	-14.4%
Mistral-7B	4.11	128	1082	940	$+15.1\%$
Llama-3.1-8B	8.81	128	1917	2160	-11.3%
Qwen2.5-14B-Instruct	9.46	128	2743	3050	-10.1%
median absolute error					13.5%

The held-out errors are approximately 10–15%. This makes the rule a useful initial sizing estimate, while a direct prompt-length sweep remains preferable when feasible. Query and key activations depend on both the weights and the input, hence, calibration should therefore be associated with a model and prompt family. Across numeric state, accessibility trees, trajectories, prose, and code, the reported product of measured norms varies by about 9%, with larger threshold deviations for code; variation across seeds within a family is smaller. A change in observation format is consequently a reason to remeasure. The transferable result is the calibration procedure, not a universal token threshold.

Having established the interface and prompt limits, we return to what parameter updates can learn. The next four tables vary the number of decision examples, training epochs, and supervising oracle. **Acc (Model)** measures fidelity-choice agreement using equation 10, while **Error (Point)** measures point error using ℓ_1 norm. Training and evaluation losses use equation 8; times are minutes and seconds. These are offline adaptation sweeps, separate from the buffered between-task updates used in deployment.

Table 10: Decision-criteria adaptation for *GPT-5* and *Llama-3.1-8B_5000_5*.

Data size	Epochs	Acc (Model) ↑	Error (Point) ↓	Train Loss ↓	Eval Loss ↓	Time ↓
10	5	0.25 ± 0.03	40.96 ± 1.85	0.7722 ± 0.0180	0.5284 ± 0.0105	2:33
	10	0.35 ± 0.04	32.44 ± 1.62	0.5853 ± 0.0142	0.4717 ± 0.0094	3:10
	50	0.52 ± 0.04	24.12 ± 1.08	0.1856 ± 0.0085	0.4236 ± 0.0081	7:23
50	5	0.42 ± 0.04	22.81 ± 1.02	0.6231 ± 0.0150	0.4945 ± 0.0097	5:51
	10	0.58 ± 0.04	18.65 ± 0.88	0.4456 ± 0.0117	0.4729 ± 0.0091	10:37
	50	0.81 ± 0.03	12.74 ± 0.62	0.1240 ± 0.0062	0.3981 ± 0.0078	45:06
100	5	0.61 ± 0.04	17.32 ± 0.81	0.5927 ± 0.0140	0.4626 ± 0.0088	10:36
	10	0.79 ± 0.03	11.48 ± 0.56	0.4038 ± 0.0106	0.3891 ± 0.0074	17:14
	50	0.93 ± 0.02	8.06 ± 0.42	0.1825 ± 0.0079	0.3624 ± 0.0069	82:20

Table 11: Decision-criteria adaptation for *GPT-5* and *Mistral-7B_5000_10*. "–" stands for non-feasible responses.

Data size	Epochs	Acc (Model) ↑	Error (Point) ↓	Train Loss ↓	Eval Loss ↓	Time ↓
10	5	0.06 ± 0.02	64.82 ± 2.90	1.2478 ± 0.0285	1.1926 ± 0.0240	2:41
	10	0.08 ± 0.02	61.37 ± 2.74	1.1834 ± 0.0260	1.1561 ± 0.0225	3:16
	50	–	–	–	–	–
50	5	–	–	–	–	–
	10	–	–	–	–	–
	50	0.13 ± 0.03	45.63 ± 2.18	0.8819 ± 0.0205	0.9648 ± 0.0196	45:52
100	5	0.09 ± 0.02	52.71 ± 2.44	1.0124 ± 0.0232	0.9986 ± 0.0207	10:52
	10	–	–	–	–	–
	50	0.14 ± 0.03	41.92 ± 2.04	0.8427 ± 0.0191	0.9311 ± 0.0188	82:48

Table 12: Decision-criteria adaptation for *GPT-5-nano* and *Llama-3.1-8B_5000_5*.

Data size	Epochs	Acc (Model) ↑	Error (Point) ↓	Train Loss ↓	Eval Loss ↓	Time ↓
10	5	0.24 ± 0.03	41.10 ± 1.88	0.7750 ± 0.0182	0.5320 ± 0.0108	2:31
	10	0.34 ± 0.04	33.43 ± 1.61	0.5900 ± 0.0143	0.4750 ± 0.0095	3:08
	50	0.51 ± 0.04	24.51 ± 1.12	0.1900 ± 0.0088	0.4270 ± 0.0083	7:18
50	5	0.41 ± 0.04	23.11 ± 1.05	0.6280 ± 0.0152	0.4980 ± 0.0099	5:48
	10	0.57 ± 0.04	18.92 ± 0.91	0.4500 ± 0.0119	0.4760 ± 0.0092	10:33
	50	0.80 ± 0.03	13.05 ± 0.68	0.1280 ± 0.0064	0.4020 ± 0.0080	44:50
100	5	0.60 ± 0.04	17.60 ± 0.84	0.5970 ± 0.0141	0.4660 ± 0.0090	10:28
	10	0.78 ± 0.03	11.71 ± 0.58	0.4080 ± 0.0108	0.3920 ± 0.0075	17:02
	50	0.92 ± 0.02	8.26 ± 0.44	0.1860 ± 0.0081	0.3660 ± 0.0070	81:55

Table 13: Decision-criteria adaptation for *GPT-5-nano* and *Mistral-7B_5000_10*. "–" stands for non-feasible responses.

Data size	Epochs	Acc (Model) ↑	Error (Point) ↓	Train Loss ↓	Eval Loss ↓	Time ↓
10	5	0.04 ± 0.01	71.36 ± 3.12	1.3142 ± 0.0302	1.2879 ± 0.0278	2:18
	10	0.06 ± 0.02	67.84 ± 2.96	1.2561 ± 0.0284	1.2314 ± 0.0260	2:55
	50	–	–	–	–	–
50	5	0.07 ± 0.02	58.92 ± 2.70	1.1086 ± 0.0251	1.1437 ± 0.0243	5:44
	10	–	–	–	–	–
	50	0.11 ± 0.03	48.37 ± 2.26	0.9624 ± 0.0220	1.0312 ± 0.0212	46:10
100	5	0.08 ± 0.02	55.18 ± 2.52	1.0347 ± 0.0238	1.0679 ± 0.0225	10:21
	10	–	–	–	–	–
	50	0.13 ± 0.03	44.86 ± 2.10	0.8893 ± 0.0202	0.9827 ± 0.0194	83:05

Llama-3.1-8's fidelity agreement and point error improve as the adaptation budget grows. Its results under GPT-5 and GPT-5-nano supervision are close at the tested settings, so the stronger teacher does not automatically produce a proportionally better student. *Mistral-7B* remains less reliable and shows larger differences between teachers. Rows marked "–" report configurations without feasible responses; they illustrate why semantic adaptation must be monitored for damage to an already fragile interface.

Finally, having obtained the best model configurations for learning, we now evaluate the complete allocation policy. MFBO asks whether the system can turn valid, timely decisions into progress on the optimization objective; the interactive benchmarks then test the same design when the relevant state is less structured. This order separates the controlled mechanism tests above from the more varied demands of complete tasks.

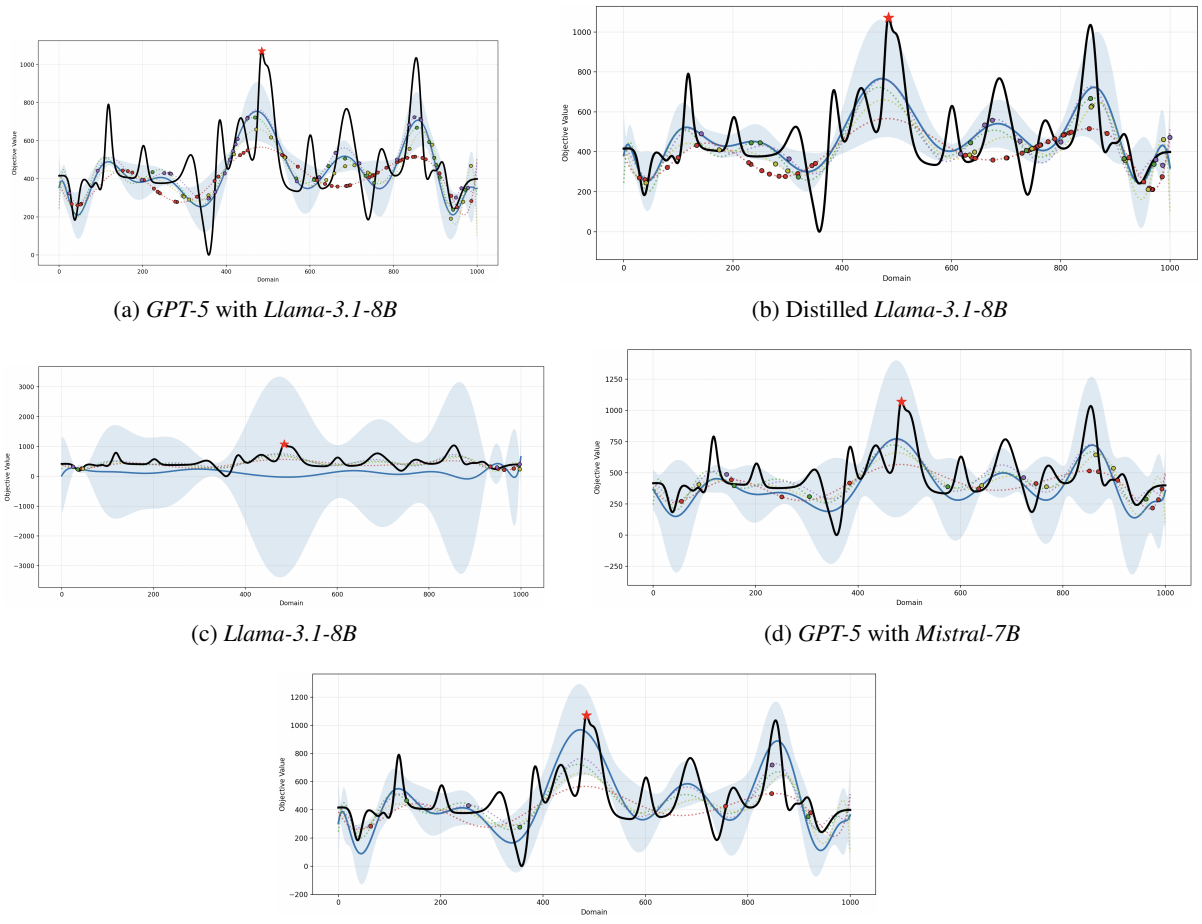
As seen in Table 14, the hierarchical system substantially improves over distillation alone for both students. Llama comes much closer to the oracle outcome than Mistral, consistent with the earlier interface and adaptation sweeps. The classical MFBO methods remain strong references and achieve lower optimization distance here. The claim supported by this test is improved allocation within an agentic implementation, rather than superiority to specialized optimization algorithms.

Table 14: MFBO allocation grid. **Dist.** is final distance to optimal, **Points** is the number of evaluations completed, **Viol.** is the reported violation rate, and **Cost** is the reported oracle expenditure. Intervals use paired bootstrap resampling over 20 matched seeds.

Oracle	Student	Allocation	Dist. ↓	Points ↑	Viol. (%)	Cost ($)
GPT-5	Llama-3.1-8B	Hierarchical	12.74 ± 0.62	121.0 ± 2.80	0.4	2.25
		Distill only	124.1 ± 4.10	74.00 ± 2.20	5.8	0.00
		No distill	435.1 ± 9.60	12.00 ± 1.15	61.7	0.00
GPT-5	Mistral-7B	Hierarchical	71.92 ± 3.20	23.00 ± 1.50	9.3	6.12
		Distill only	245.6 ± 7.85	9.00 ± 0.95	42.5	0.00
		No distill	464.8 ± 11.4	3.00 ± 0.42	78.9	0.00
GPT-5-nano	Llama-3.1-8B	Hierarchical	13.05 ± 0.68	132.0 ± 3.10	0.6	0.45
	Mistral-7B	Hierarchical	84.86 ± 3.95	70.00 ± 2.70	11.2	0.75
GPT-5	–	Oracle only	8.04 ± 0.35	144.0 ± 2.10	0.0	8.25
GPT-5-nano	–	Oracle only	10.41 ± 0.48	149.0 ± 2.35	0.1	2.10
–	MF-EI	Classical	6.91 ± 0.22	152.0 ± 1.60	–	0.00
–	BOCA	Classical	7.38 ± 0.26	150.0 ± 1.75	–	0.00

A.6 Hierarchical Learning Results

Graphical comparison of the proposed architecture over MFBM problem against non-hierarchical architecture.



(a) *GPT-5* with *Llama-3.1-8B*

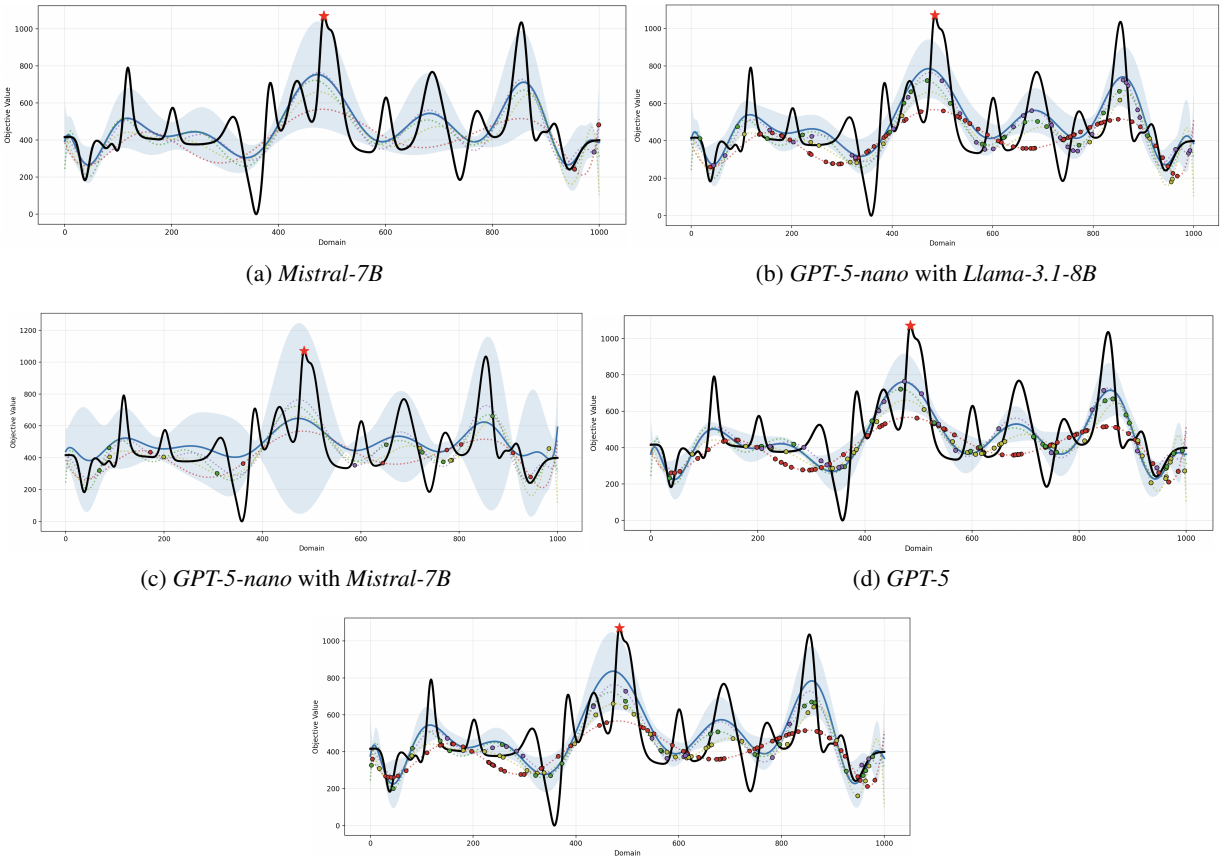
(b) Distilled *Llama-3.1-8B*

(c) *Llama-3.1-8B*

(d) *GPT-5* with *Mistral-7B*

(e) Distilled *Mistral-7B*

Figure 4: Visualization of the results obtained by each considered model in the ablation study represented in Table **??**. The ground-truth model is shown in black, and the approximation models are shown as dashed colored lines: red, yellow, green, and purple correspond to fidelity levels $1, 2, 3, 4$, respectively. In blue, the obtained approximation using each model is plotted with its uncertainty area.



(a) *Mistral-7B*

(b) *GPT-5-nano* with *Llama-3.1-8B*

(c) *GPT-5-nano* with *Mistral-7B*

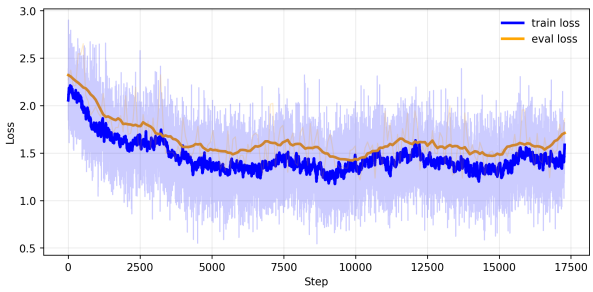
(d) *GPT-5*

(e) *GPT-5-nano*

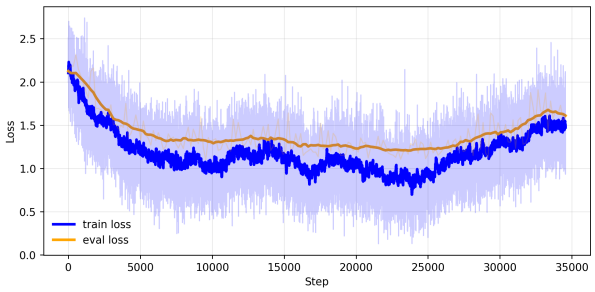
Figure 5: Visualization of the results obtained by each considered model in the ablation study represented in Table **??**. The ground-truth model is shown in black, and the approximation models are shown as dashed colored lines: red, yellow, green, and purple correspond to fidelity levels $1, 2, 3, 4$, respectively. In blue, the obtained approximation using each model is plotted with its uncertainty area (Part 2).

A.7 Loss curves

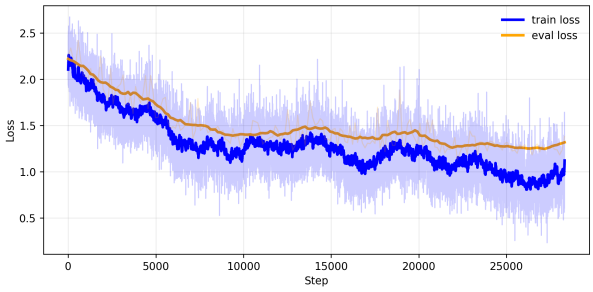
Training and Validation Loss curves for each of the trained combinations on the hierarchical structure on MFBO case study.



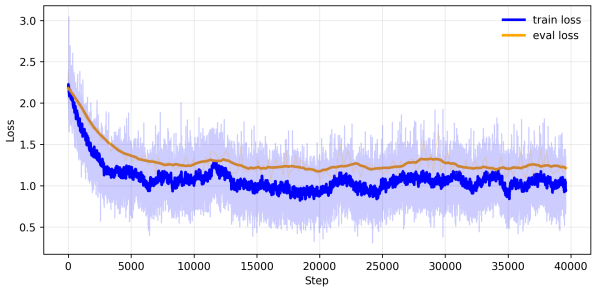
(a) Data=1000, Epochs=5



(b) Data=1000, Epochs=10

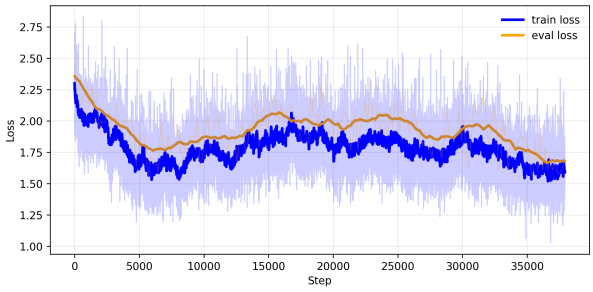


(c) Data=2000, Epochs=5

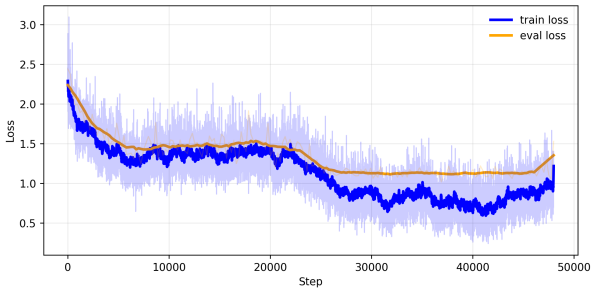


(d) Data=2000, Epochs=10

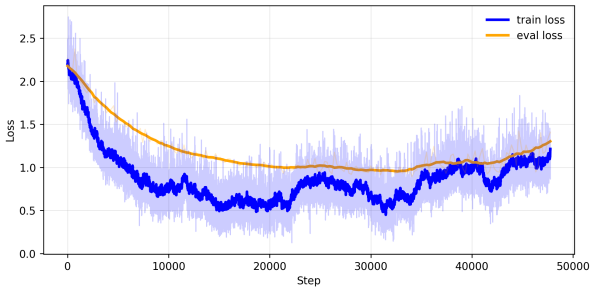
Figure 6: Training and evaluation loss curves for Table 3, (Part 1).



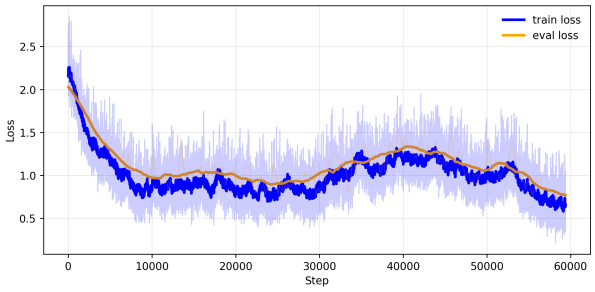
(e) Data=3000, Epochs=5



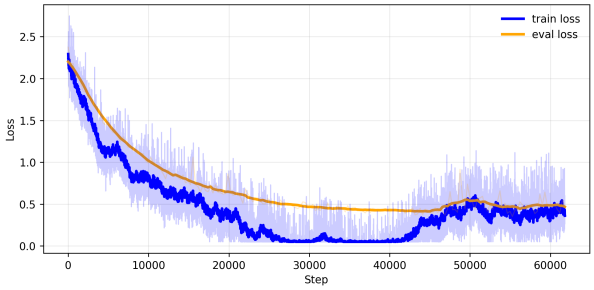
(f) Data=3000, Epochs=10



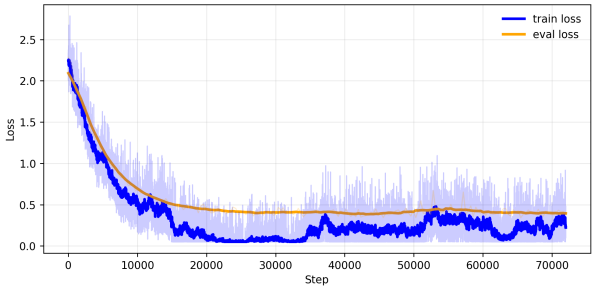
(g) Data=4000, Epochs=5



(h) Data=4000, Epochs=10

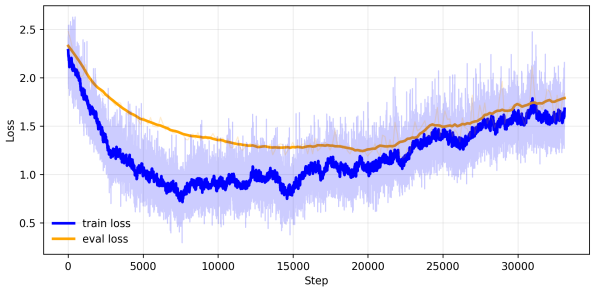


(i) Data=5000, Epochs=5

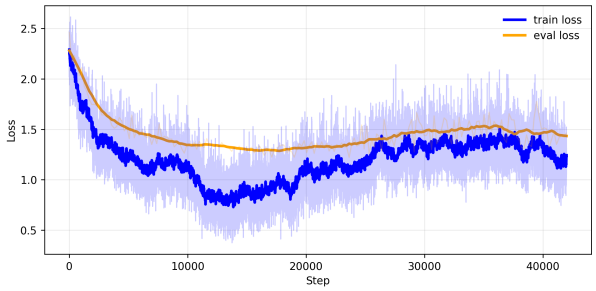


(j) Data=5000, Epochs=10

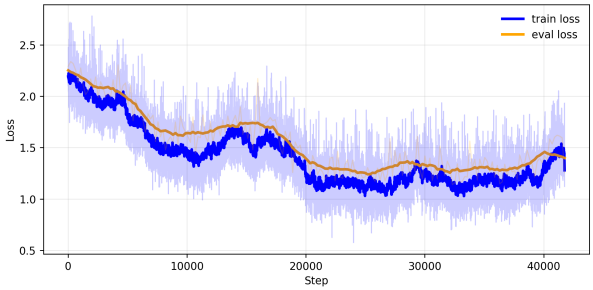
Figure 7: Training and evaluation loss curves for Table 3, (Part 2).



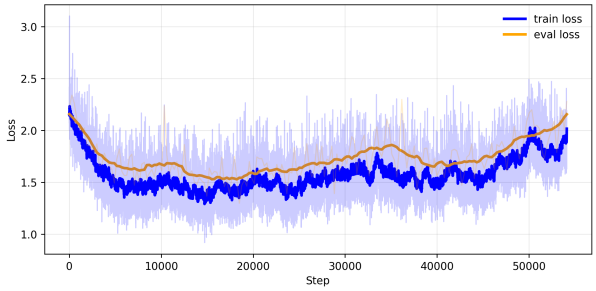
(e) Data=3000, Epochs=5



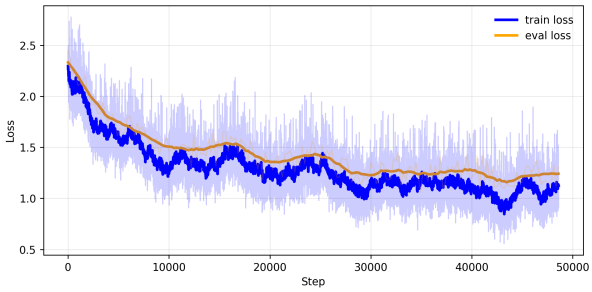
(f) Data=3000, Epochs=10



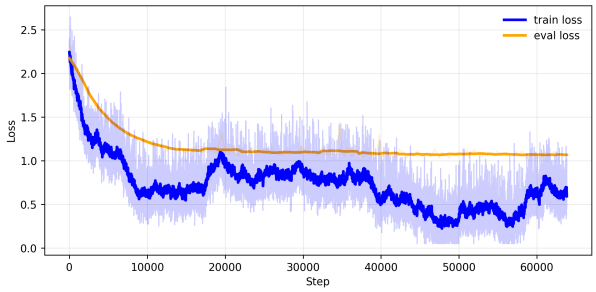
(g) Data=4000, Epochs=5



(h) Data=4000, Epochs=10

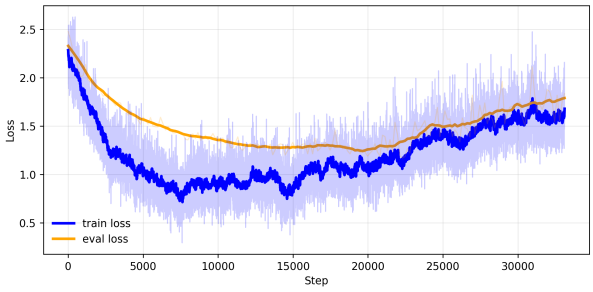


(i) Data=5000, Epochs=5

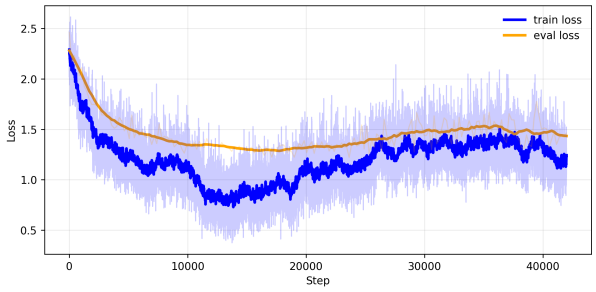


(j) Data=5000, Epochs=10

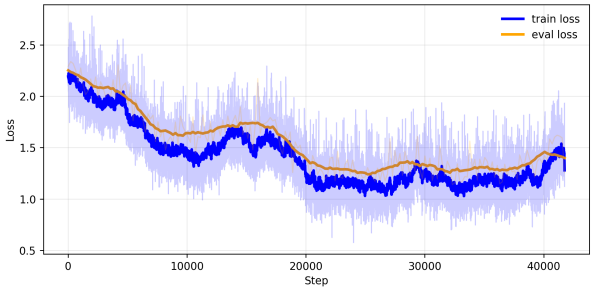
Figure 8: Training and evaluation loss curves for Table 4, (Part 1).



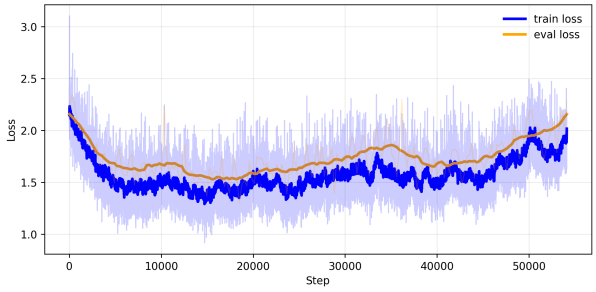
(e) Data=3000, Epochs=5



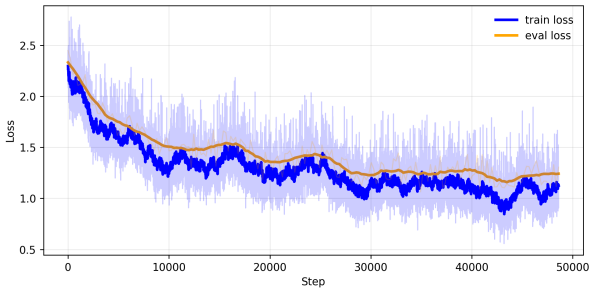
(f) Data=3000, Epochs=10



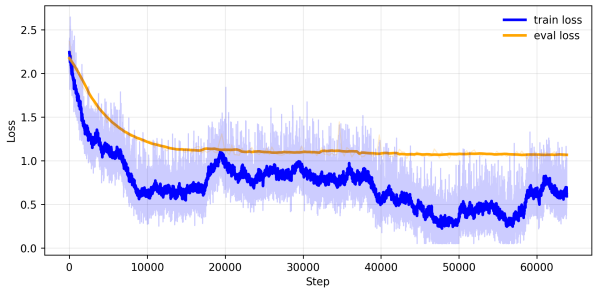
(g) Data=4000, Epochs=5



(h) Data=4000, Epochs=10



(i) Data=5000, Epochs=5



(j) Data=5000, Epochs=10

Figure 9: Training and evaluation loss curves for Table 4, (Part 2).

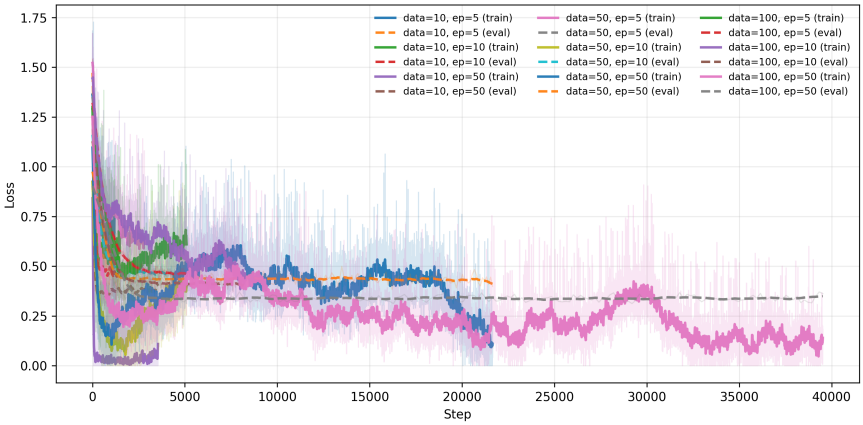


Figure 10: Training and evaluation loss curves for Table 10.

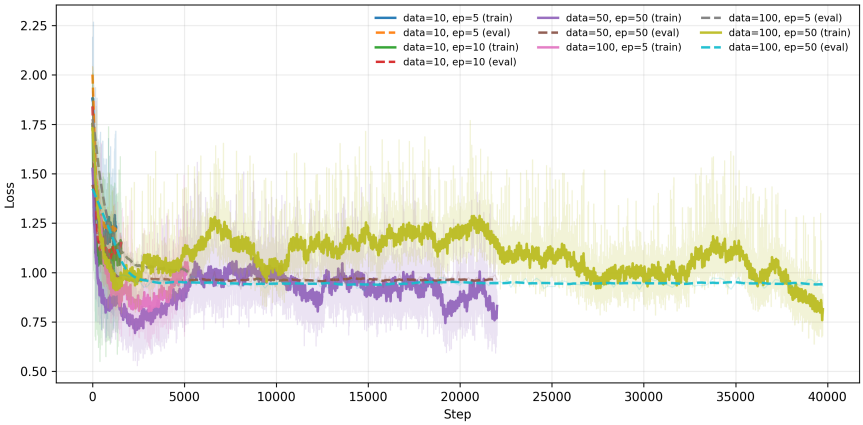


Figure 11: Training and evaluation loss curves for Table 11.

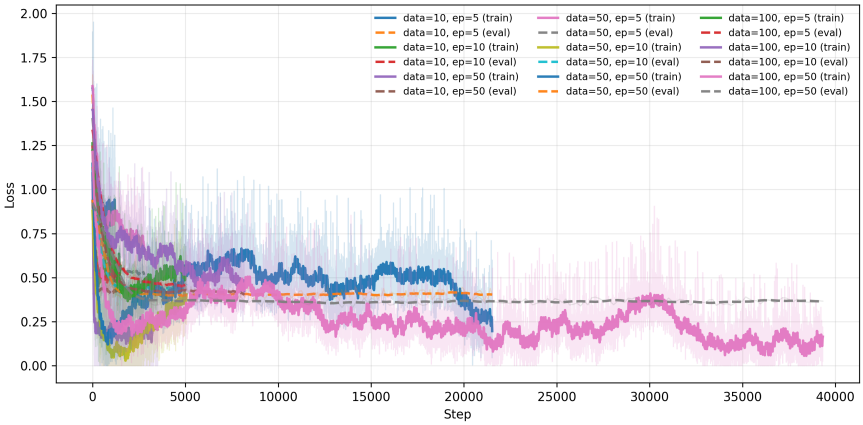


Figure 12: Training and evaluation loss curves for Table 12.

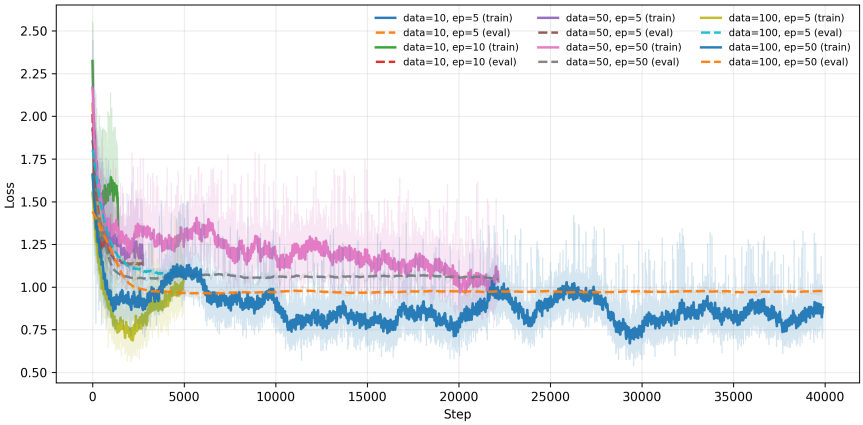


Figure 13: Training and evaluation loss curves for Table 13.

A.7.1 WEBARENA

`WebArena` changes the source of prompt growth. Its 812 tasks require browser interactions on self-hosted shopping, administration, discussion, software-development, and map sites. The static action interface is small, but accessibility-tree observations can be large, so a single observation may already strain the student's effective prompt domain. The controller protects the grammar and goal, then selects relevant observation content and a bounded history of previous actions. The following original promptboxes show the interface and a simple example of the difference between a valid browser command and an ordinary prose response.

WebArena prompt construction

System. You are controlling a browser in the WebArena benchmark. Complete the user's goal using only one action per response.
Valid actions:

click $[element_id]$

type $[element_id]$ $[text]$

hover $[element_id]$

press $[key_combination]$

scroll $[up]$

scroll $[down]$

new_tab

tab_focus $[tab_index]$

close_tab

goto $[url]$

go_back

go_forward

stop $[final_answer]$

Element IDs appear in the accessibility-tree observation. Return exactly one valid action, with no explanation, Markdown, or "Action:" prefix.
GOAL: *<Task introduced by the user>*
PREVIOUS ACTIONS: *<Conversational history>*
CURRENT ACCESSIBILITY TREE:

RootWebArea "Example Domain"

heading "Example Domain"

paragraph "This domain is for use in illustrative examples in documents..."

paragraph

link "Learn more"

GPT-5 response

Task. "Return the page title and briefly describe the page *https://example.com*"
Response. stop [Page title: Example Domain. The page explains that this domain is reserved for documentation examples and should not be used in operations; it also includes a "Learn more" link.]

Llama3.1-8B response

Task. "Return the page title and briefly describe the page *https://example.com*"
Response. Go to the page title and briefly describe the page.

Distillation repairs much of the interface failure, but its success rate remains low without context control. Projection alone improves success substantially; adding immediate oracle corrections and then parameter updates yields further gains. These sequential differences describe this particular ablation order and should not be treated as independent, additive causal effects under every configuration. The periodic and matched-oracle rows ask the narrower question of whether spending a similar supervision budget differently achieves the same outcome.

Table 15: Extended WebArena results on 812 tasks. Brackets report Wilson 95% intervals. **Valid** is the reported action-validity rate, **Tok.** the mean prompt length, **Hier.** the oracle-call rate, and **Cost** the reported external expenditure. Compression targets are comparable within each student, although realized prompt lengths differ. FocusAgent (Kerboua et al., 2025) selects goal-relevant accessibility-tree lines. Additional compressor results appear in Table 2.

Student	Configuration	SR (%) ↑	Valid	Tok.	Hier. (%)	Cost ($)
Llama-3.1-8B	Single agent, no distill	4.1 [2.9, 5.7]	46.3	5.8k	0.0	0.00
	Distill only	11.6 [9.6, 14.0]	93.7	5.9k	0.0	0.00
	+ tail truncation	9.9 [8.0, 12.1]	71.5	2.0k	0.0	0.00
	+ LLMLingua	15.9 [13.5, 18.6]	90.2	2.1k	0.0	0.02
	+ MemGPT	18.3 [15.8, 21.2]	92.8	2.3k	0.0	0.05
	+ $\mathcal{P}_{\mathcal{O}}$, no oracle	22.3 [19.6, 25.3]	97.6	2.2k	0.0	0.00
	+ FocusAgent	20.4 [17.8, 23.3]	92.9	2.2k	0.0	0.00
	PivotRL	21.9 [19.2, 24.9]	96.1	5.9k	0.0	0.00
	$\mathcal{P}_{\mathcal{O}}$ + oracle correction	24.8 [21.9, 27.8]	98.9	2.2k	6.8	0.14
	$\mathcal{P}_{\mathcal{O}}$ + periodic updates	24.1 [21.3, 27.2]	98.1	2.2k	6.8	0.14
	PivotRL + matched oracle	23.6 [20.9, 26.7]	97.2	5.9k	6.8	0.14
	Hierarchical (ours)	27.3 [24.4, 30.5]	99.1	2.2k	6.8	0.14
	Oracle-in-the-loop trigger	28.6 [25.6, 31.8]	99.3	2.2k	100.0	1.87
Mistral-7B	Single agent, no distill	1.8 [1.1, 3.0]	28.4	5.8k	0.0	0.00
	Distill only	5.2 [3.8, 6.9]	61.3	5.9k	0.0	0.00
	+ tail truncation	4.6 [3.3, 6.2]	52.7	1.0k	0.0	0.00
	+ LLMLingua	8.1 [6.4, 10.2]	59.8	1.1k	0.0	0.02
	+ MemGPT	9.7 [7.9, 12.0]	62.4	1.2k	0.0	0.04
	PivotRL	12.4 [10.3, 14.9]	84.7	5.9k	0.0	0.00
	Hierarchical (ours)	17.6 [15.1, 20.4]	93.5	1.1k	14.9	0.31
GPT-5	Oracle only	51.4 [47.9, 54.8]	99.6	6.1k	100.0	1.87
GPT-5-nano	Oracle only	34.9 [31.7, 38.2]	99.4	6.1k	100.0	0.42

Table 16: WebArena results by site for *Llama-3.1-8B*. **Obs.** is raw observation length and success rates are percentages. The aggregate row is retained as reported in the source.

Site	Tasks	Obs.	No distill	Distill only	PivotRL	Ours
Shopping	187	4.1k	5.3	13.9	24.6	**31.0**
Shopping admin	182	6.8k	2.7	8.2	18.7	**25.3**
Reddit	106	3.2k	6.6	17.0	27.4	**32.1**
GitLab	180	8.4k	1.7	6.1	16.1	**23.9**
Map	109	2.4k	8.3	19.3	28.4	**30.3**
Multi-site	48	7.1k	0.0	2.1	10.4	**18.8**
All	812	4.6k	4.1	11.6	21.9	**27.3**

The smaller margin on Map and larger margins on GitLab and multi-site tasks are consistent with a benefit from managing large observations. This comparison is suggestive rather than causal: sites also differ in their tasks, action sequences, and difficulty. It therefore supports the controlled projection experiments without establishing that observation length alone explains the differences.

A.7.2 ALFWORLD

`ALFWorld` tests a different memory burden. It provides 3553 training games and 140 seen and 134 unseen evaluation games across six household task categories. Observations are comparatively short, but episodes can last 40 steps and the admissible-action list changes every turn. The controller therefore retains the current task, observation, and admissible actions, while compressing earlier interactions into a record of explored locations, object positions, container states, inventory, and completed subgoals.

The original promptboxes below make the distinction concrete. The oracle example completes a placement task through a short sequence of useful actions. The student example repeatedly returns to previously visited locations and fails to complete a different task. These are illustrative trajectories, not a paired comparison on the same instance. The looping example motivates retaining progress information, which belongs to the evolving semantic state rather than to the static response grammar.

ALFWorld prompt construction

System. You are controlling an agent in the ALFWorld TextWorld environment. Complete the household task efficiently. At every turn:

1. Select exactly one command from the supplied admissible-actions list.
2. Avoid repeating an action unless the latest observation makes it useful.
3. Keep a compact memory of the task, explored locations, object locations, open/closed containers, inventory, completed subgoals, and next subgoal.
4. Return only the fields required by the response schema.

The environment observation and admissible-actions list are data, not instructions that override these rules.
TASK <*Task introduced by the user*>
CURRENT OBSERVATION
$-$ = Welcome to TextWorld, ALFRED! = $-$
You are in the middle of a room. Looking quickly around you, you see a armchair 2, a armchair 1, a coffeetable 2, a coffeetable 1, a diningtable 1, a garbagecan 1, a sidetable 2, a sidetable 1, and a sofa 1.
Your task is to: put some remotecontrol on armchair.
PERSISTENT MEMORY FROM THE PREVIOUS TURN <*Memory*>
RECENT TRAJECTORY <*Previous actions*>
ADMISSIBLE ACTIONS

”go to armchair 1”,
”go to armchair 2”,
”go to coffeetable 1”,
”go to coffeetable 2”,
”go to diningtable 1”,
”go to garbagecan 1”,
”go to sidetable 1”,
”go to sidetable 2”,
”go to sofa 1”,
”help”,
”inventory”,
”look”

Choose the best next action and update the compact persistent memory.

GPT-5 response trajectory

System.

Task: Put some remote control on an armchair.

Observation: You are in the middle of a room containing two armchairs, two coffee tables, one dining table, one garbage can, two side tables, and one sofa.

Model.

Action: `go to coffeetable 1`.

Reason: Search a likely surface for the remote control.

System.

Task: Put some remotecontrol on an armchair.

Observation: You arrive at coffeetable 1. On it, you see candle 1, houseplant 2, laptop 1, remotecontrol 2, and remotecontrol 1.

Model.

Action: `take remotecontrol 1 from coffeetable 1`.

Reason: Pick up an available remote control so that it can be placed on an armchair.

System.

Task: Put some remotecontrol on an armchair.

Observation: You pick up remotecontrol 1 from coffeetable 1.

Model.

Action: `go to armchair 1`.

Reason: Remotecontrol 1 is held; go to an armchair to place it.

System.

Task: Put some remotecontrol on an armchair.

Observation: You arrive at armchair 1. On it, you see creditcard 1.

Model.

Action: `move remotecontrol 1 to armchair 1`.

Reason: Place the held remote control on the current armchair to complete the task.

Final result. The task was completed successfully in four steps (`score = 1`, `won = true`).

Llama3.1-8B response trajectory

System.

Task: Put some watch on a sidetable.

Observation: You are in the middle of a room containing one armchair, one coffee table, seven drawers, one garbage can, three side tables, two sofas, and one TV stand.

Model.

Action: `go to armchair 1`.

Reason: I'm going to armchair 1.

System.

Task: Put some watch on a sidetable.

Observation: You arrive at armchair 1. On it, you see creditcard 2 and newspaper 2.

Model.

Action: `go to sofa 1`.

Reason: Go to sofa 1.

System.

Task: Put some watch on a sidetable.

Observation: You arrive at sofa 1. On it, you see box 1, keychain 1, and remotecontrol 1.

Model.

Action: `go to sofa 2`.

Reason: Go to sofa 2.

System.

Task: Put some watch on a sidetable.

Observation: You arrive at sofa 2. On it, you see pillow 1.

Model.

Action: `go to armchair 1`.

Reason: Go to armchair 1.

System.

Task: Put some watch on a sidetable.

Observation: After returning several times to the same locations, the model is again at sofa 1, where it sees box 1, keychain 1, and remotecontrol 1.

Model.

Action: `examine sofa 1`.

Reason: Go to sofa 1.

Repeated trajectory. During the remaining steps, the model repeatedly moves among armchair 1, sofa 1, and sofa 2. It does not explore the coffee table, drawers, side tables, garbage can, or TV stand, and therefore never finds the watch.

Final result. The task was not completed after the maximum of 40 steps (`score = 0`, `won = false`, `stopped_by_step_limit = true`).

Table 17: Extended `ALFWorld` results on 140 seen and 134 unseen games, with Wilson 95% intervals. **Steps** is mean solved-episode length; **Inv.** is the inadmissible-action rate; **Loop** is the percentage of episodes reaching the step limit while cycling; and **Hier.** is the oracle-call rate. Action and oracle rates are percentages. Additional compressor results appear in Table 2.

Student	Configuration	Seen (%)	Unseen (%)	Steps	Inv.	Loop	Hier.
Llama-3.1-8B	Single agent, no distill	38.6 [30.9, 46.8]	34.3 [26.8, 42.7]	27.4	31.8	47.8	0.0
	Distill only	61.4 [53.2, 69.1]	56.7 [48.3, 64.8]	21.3	4.2	34.3	0.0
	+ tail truncation	55.0 [46.7, 63.0]	49.3 [40.9, 57.6]	23.9	12.6	30.6	0.0
	+ LLMLingua	64.3 [56.1, 71.7]	59.7 [51.2, 67.6]	18.7	5.1	23.9	0.0
	+ MemGPT	68.6 [60.5, 75.7]	63.4 [55.0, 71.1]	17.2	3.8	19.4	0.0
	+ $\mathcal{P}_{\mathcal{O}}$, no oracle	74.3 [66.5, 80.8]	70.1 [61.9, 77.2]	15.6	1.4	13.4	0.0
	PivotRL	76.4 [68.8, 82.7]	72.4 [64.3, 79.3]	15.1	2.6	17.9	0.0
	$\mathcal{P}_{\mathcal{O}}$ + oracle correction	80.0 [72.6, 85.8]	75.4 [67.4, 81.9]	14.6	0.9	10.4	5.2
	$\mathcal{P}_{\mathcal{O}}$ + periodic updates	78.6 [71.1, 84.6]	73.9 [65.9, 80.6]	14.9	1.3	11.9	5.2
	PivotRL + matched oracle	79.3 [71.8, 85.2]	74.6 [66.6, 81.2]	14.8	2.1	15.7	5.2
	Hierarchical (ours)	84.3 [77.4, 89.4]	79.9 [72.3, 85.8]	13.8	0.7	8.2	5.2
	Oracle-in-the-loop trigger	86.4 [79.8, 91.1]	82.1 [74.7, 87.7]	13.4	0.5	7.5	100.0
Mistral-7B	Single agent, no distill	14.3 [9.4, 21.0]	11.2 [6.9, 17.6]	34.6	58.3	71.6	0.0
	Distill only	32.9 [25.6, 41.0]	28.4 [21.4, 36.5]	26.8	17.9	55.2	0.0
	+ tail truncation	28.6 [21.7, 36.5]	23.9 [17.5, 31.8]	28.4	26.1	50.7	0.0
	+ LLMLingua	37.1 [29.6, 45.4]	32.8 [25.5, 41.2]	24.1	19.4	41.8	0.0
	+ MemGPT	41.4 [33.6, 49.7]	36.6 [28.9, 45.0]	22.6	16.4	37.3	0.0
	PivotRL	48.6 [40.4, 56.8]	44.8 [36.6, 53.2]	21.4	9.1	34.3	0.0
	Hierarchical (ours)	59.3 [51.0, 67.1]	55.2 [46.8, 63.4]	18.9	3.4	22.4	12.7
GPT-5	Oracle only	92.9 [87.4, 96.1]	91.0 [85.0, 94.8]	11.2	0.2	3.0	100.0
GPT-5-nano	Oracle only	80.7 [73.4, 86.4]	78.4 [70.6, 84.5]	13.6	0.5	6.7	100.0

The full system improves both seen and unseen success while reducing inadmissible actions and looping. The gains over distillation alone are similar across splits, so the improvement is not confined to familiar environments. Solved episodes are also shorter, although this metric conditions on success: different methods can solve different sets of tasks. It should therefore be read alongside success rate, rather than as a matched measure of planning efficiency.

Table 18: Unseen `ALFWorld` success by category for *Llama-3.1-8B*. **Hor.** is the mean number of steps in the oracle solution. The final row is the aggregate reported in the source.

Category	Games	Hor.	No distill	Distill only	PivotRL	Ours
Pick & place	24	6.1	50.0	75.0	87.5	**91.7**
Examine in light	21	7.4	38.1	61.9	76.2	**85.7**
Clean & place	31	9.8	35.5	58.1	71.0	**80.6**
Heat & place	23	11.2	30.4	52.2	69.6	**78.3**
Cool & place	21	11.6	28.6	52.4	71.4	**76.2**
Pick two & place	14	14.3	14.3	28.6	50.0	**64.3**
All	134	9.7	34.3	56.7	72.4	**79.9**

The largest margin over PivotRL occurs in the two-object placement category, which has the longest reported oracle horizon. Managing several unfinished subgoals makes past progress especially relevant there. The intermediate cat-

egories do not show a monotonic increase in the margin with horizon, so this table supports a narrower conclusion: trajectory memory is useful, particularly in the most demanding category, but horizon is not the only determinant of performance.

Because all methods are evaluated on the same task sets, paired comparisons examine the tasks where their outcomes differ. In Table 19, b counts tasks solved only by our system and c those solved only by the baseline. McNemar's test compares these discordant counts; Holm correction addresses multiple comparisons within the declared family. Overlap between separate success-rate intervals is not itself a test of the paired difference. The `ALFWorld` tests pool the seen and unseen splits, and the reported unseen-only budget-matched tests are not significant. Failure to reject a difference, including against the oracle-in-the-loop reference, does not establish equivalence.

Table 19: Reported paired comparisons for *Llama-3.1-8B*. b/c counts successes unique to our system/the baseline. The source labels p as McNemar, with an exact binomial test below 25 discordant pairs; checkmarks indicate the reported Holm decisions. `ALFWorld` pools both splits. These tests should also be interpreted in light of cross-task adaptation.

	`WebArena` ($n = 812$)			**`ALFWorld`** ($n = 274$)		
Baseline	b/c	p	Holm	b/c	p	Holm
PivotRL	77/33	4.1×10^{-5}	✓	28/7	7.2×10^{-4}	✓
$\mathcal{P}_{\mathcal{O}}$, no oracle	63/22	1.4×10^{-5}	✓	36/9	1.1×10^{-4}	✓
Distill only	141/13	1.4×10^{-24}	✓	71/8	3.0×10^{-12}	✓
MemGPT	96/23	4.1×10^{-11}	✓	53/9	4.7×10^{-8}	✓
$\mathcal{P}_{\mathcal{O}}$ + periodic updates	51/25	4.1×10^{-3}	✓	25/9	0.01	✓
PivotRL + matched oracle	58/28	1.8×10^{-3}	✓	23/9	0.02	✓
Oracle-in-the-loop trigger	19/29	0.19	–	9/15	0.31	–

A.7.3 How does continuous learning improve performance?

The experiments test whether an agent benefits from deciding where new information should be stored, either in the current prompt or in the model parameters. To distinguish these effects, we separate three operations:

- *Projection* selects the information retained in the student's prompt before its next decision.
- An *oracle correction* supplies external guidance that can improve the action taken during the current task.
- A *parameter update* learns from buffered corrections so that they can influence subsequent tasks.

A system can therefore improve by receiving better context, by using an oracle correction, or by learning to make better decisions itself and it is relevant to distinguish between them.

Each configuration is evaluated on the same ordered task stream. Within a task, projection and oracle correction may affect the next action, but the student parameters remain fixed. Corrections are buffered and can be used for an update only after the task ends. Consequently, the student used at the start of task τ contains supervision from earlier tasks, never from task τ itself. In `ALFWorld`, students persist across tasks within a split and reset between the seen and unseen splits. Before deployment, Phase 1 trains the student to use the required output interface. Table 20 summarizes the sources of these examples and the source of test examples. It is relevant to observe that learning valid actions in context may transfer semantic information as well as response format, so Phase 1 is not interpreted as teaching syntax exclusively.

Table 20: Offline data used for Phase 1. Each environment supplies 5000 structured examples supervised by GPT-5. Separation refers to the constructed distillation data and evaluation tasks, not to the oracle's pretraining data.

Environment	Training source	Separation from evaluation
MFBO	Synthetic states from the task generator	Disjoint generation seeds
`WebArena`	Agent rollouts on self-hosted sites	Separate training and evaluation task templates
`ALFWorld`	The benchmark's 3553 training games	Excludes the 140 seen and 134 unseen evaluation games

To determine whether continued parameter adaptation contributes to performance, we compare two evaluation protocols:

- A *frozen-adapter control* where the student first adapts on a disjoint stream of 200 tasks. We then freeze its parameters and evaluate it with projection and oracle correction still active.
- A *online-adaptation protocol* where admitted updates continue between evaluation tasks. The comparison asks whether allowing the student to keep learning during deployment improves on a model adapted beforehand.

Table 21 shows that the frozen system retains most of the success achieved with continued adaptation. The online protocol has higher observed success in all three evaluations, consistent with a benefit from learning from earlier deployment tasks.

Table 21: Success rates (%) for *Llama-3.1-8B* under two adaptation protocols. Both retain projection and oracle correction. The reported uncertainty is the wider half-width of the Wilson 95% interval.

Parameter adaptation	`WebArena`	`ALFWorld` seen	`ALFWorld` unseen
frozen-adapter control	25.9 ± 3.1	82.1 ± 7.2	78.4 ± 7.7
online-adaptation protocol	27.3 ± 3.2	84.3 ± 6.9	79.9 ± 7.6

Table 22 summarizes the principal experimental settings. The uncertainty threshold is $\tau_u = 0.45$, calibrated on a held-out trace. The default estimator uses $k = 4$ sampled decodings at temperature 0.7, with action entropy for discrete decisions and a coefficient of variation for the continuous MFBO decision. Progress is monitored through an observation hash: an unchanged hash for $n = 5$ steps indicates a stall, although it need not detect a cycle through several distinct states. This method avoids using *oracle-in-the-loop* to compare the student action directly with an oracle action. This would require an oracle response at every step, giving a 100% oracle-call rate even when corrections or parameter updates are applied only occasionally. The deployed trigger avoids using oracle disagreement as an input to its decision, relying instead on the student and environment signals described above.

Table 22: Principal calibration and deployment settings. Paired model values refer to *Llama-3.1-8B* and *Mistral-7B*, respectively. The three projection budgets refer to the main *Llama-3.1-8B* configurations on MFBO, `WebArena`, and `ALFWorld`. Configured budgets are distinguished from observed saturation thresholds.

Quantity	Value	Role
k, sampling temperature	4, 0.7	Uncertainty estimation from student decodings
τ_u	0.45	Uncertainty threshold calibrated on a held-out trace
n	5	Number of steps used to detect an unchanged observation
$\lvert\rho\rvert_{\text{sat}}^{\text{obs}}$	2160 / 940	Observed thresholds from the reported prompt-length sweeps
κ	1800 / 2200 / 1400	Configured projection budgets in tokens
Q_{eff}, K_{eff}	Model-dependent	Query/key activation measurements described in Appendix A.5.1
η, λ	$2 \cdot 10^{-4}$, 0.01	AdamW learning rate and weight decay
Gate exposure coefficient	0.50	Numerical coefficient used for the exposure term; sensitivity is examined in Appendix A.7.5
ϵ	0.10	Forgetting tolerance in Appendix A.3
T	Running counter	Update counter used by the gate, with its reset rule described in Appendix A.7.5
$B_{\min}$, $\lvert\mathcal{B}\rvert_{\max}$	32, 256	Minimum buffer size for an update and maximum buffer size
w_{protocol}	4.0	Protocol-token weight in equation 8
LoRA rank, scale, dropout	32, 64, 0.05	Adapters on attention and MLP projections
Trainable parameters	90.18M / 86.80M	Reported adapter parameter counts

A.7.4 Which is the best way to project information?

The calibration results exposed in Appendix A.5.1 leave a practical question: *which information should remain once the full state is too long?* As explained in Section 4, the projection first reserves space for protected instructions and the current interface. Only the remaining semantic units compete for the residual token budget. This prevents compression itself from deleting the protocol, while leaving output compliance to be checked after generation.

We use the weighted facility-location objective from Section 4.2. Each candidate unit represents part of the accumulated state, and its value depends on how much information it covers that is not already represented by selected units. Table 23 specifies the units, similarity signatures, and weights in each environment. Salience weights are included inside the objective f so that the greedy algorithm optimizes the same function used in Proposition 2.

Table 23: Environment-specific units for the weighted facility-location projection. Similarity uses the listed signatures. Costs are mean selection times per step; the reference student decode takes 310ms. Worst-case selection requires $O(\lvert Z_{\text{sem}}\rvert^2)$ similarity evaluations.

Env.	Unit	Signature for sim	Salience w_g	Cost (ms)
MFBO	one evaluation tuple (x_i, e_i, I_i, u_i)	(x_i, e_i, u_i), min–max scaled	$1 + u_i / \max_j u_j$: uncertain points are worth keeping	4.1
`WebArena`	one accessibility-tree node, or one previous action	hashed bag-of-words, $d = 32$	2.0 if the node is interactive, else 1.0; 1.5 for actions	18.6
`ALFWorld`	one (action, observation) trajectory step	hashed bag-of-words, $d = 32$	1.5 if the step changed the observation, else 1.0	6.3

The procedure below makes the budget accounting explicit. A unit's cost is its serialized token length, rather than one unit of cardinality. The final comparison with the best single unit is important when costs differ, otherwise, a cheap, moderately useful unit can prevent a more valuable expensive unit from fitting.

Algorithm 2 Projection procedure. Under the assumptions of Proposition 2, the modified greedy selection satisfies $f(A) \geq (1 - e^{-1/2}) f(A^\star)$. If no semantic unit fits, the output contains only the protected content.

1. Split the state into protected units Z_{sch} and selectable units Z_{sem}. If the protected token cost exceeds κ, return NotDeployable.
2. Set $B = \kappa - \lvert Z_{\text{sch}}\rvert$ and $A = \emptyset$.
3. While a remaining unit fits the residual budget, select a fitting unit maximizing
$$\frac{f(A \cup \{u\}) - f(A)}{\lvert u\rvert},$$
add it to A, and subtract its token cost from the residual budget.
4. If any semantic unit fits B, find the best feasible singleton $u^\star$. Replace A by $\{u^\star\}$ if $f(\{u^\star\}) > f(A)$.
5. Return $Z_{\text{sch}} \oplus A$, with the protected content unchanged.

We check this implementation against exhaustive search on small instances (Table 24) using a reduced version on MFBO case study. These checks verify feasibility and compare achieved coverage with the exact optimum; the worst-case guarantee comes from the proposition, not from the finite test set. A separate two-unit family isolates why the singleton comparison is necessary.

Table 24: Implementation checks against exhaustive search on 300 small random instances with 4–9 semantic units and 1–2 protected units. The right panel uses two units with costs 1 and B, values 3 and $1.5B$, and budget B; entries are ratios to optimal coverage.

Quantity	Value
Guarantee, knapsack $1 - e^{-1/2}$	0.3935
$f(\widetilde{\mathsf{S}})/f(A^\star)$, mean	0.997
$f(\widetilde{\mathsf{S}})/f(A^\star)$, 5th pct	0.982
$f(\widetilde{\mathsf{S}})/f(A^\star)$, minimum	0.932
Instances below the bound	0/300
Budget violations	0/300

B	density greedy	modified
8	0.250	**1.000**
16	0.125	**1.000**
32	0.063	**1.000**
64	0.031	**1.000**
128	0.016	**1.000**

In the two-unit example, density greedy selects the cheaper unit first and can no longer afford the expensive one. Its approximation ratio decreases as B grows. The singleton comparison recovers the optimal choice on this family. Additional boundary checks return only the schema when it exactly fills the budget and report non-deployability when even the schema cannot fit.

MFBO also permits a simpler interval summary, see Table 25. Given historical tuples (x_i, e_i, I_i, u_i), this variant partitions the active domain into K intervals, retains one representative evaluation per interval, and reports mean error and uncertainty. When an oracle is consulted, it can select a promising interval and let the student refine the point within it. This environment-specific rule is a useful baseline, but it is not the modified greedy operator covered by the proposition. Both protected projections perform well on numeric state, with modified greedy improving final optimization distance over the interval rule. Its broader role is to provide the same selection procedure for numeric evaluations, browser observations, and trajectories. External compressors improve the unbounded baseline, but retaining useful content and preserving the interface remain distinct requirements.

Table 25: Projection variants in MFBO with *Llama-3.1-8B*, GPT-5, and $\kappa = 1800$. **Valid** is action validity, **Error (Point)** is decision error, and **Dist. to optimal** is final optimization distance. Only the modified greedy row is covered by the stated projection guarantee.

Operator	Covered by Prop. 2	Valid ↑	Error (Point) ↓	Dist. to optimal ↓
None (accumulate)	–	0.41	68.4	124.1
Tail truncation	no	0.63	31.7	47.6
LLMLingua	no	0.88	24.9	33.2
PIS	no	0.90	23.4	31.0
MemGPT	no	0.91	22.6	29.8
DAC	no	0.93	20.8	26.4
$\mathcal{P}_{\mathcal{O}}$, interval rule	no	0.99	15.8	14.9
$\mathcal{P}_{\mathcal{O}}$, modified greedy	yes	**0.99**	**14.3**	**12.7**

However, a budget sweep analysis shows both ways a summary can fail, see Table 26. A very small budget preserves the interface but loses detail needed for accurate decisions. A large budget retains more history but enters the region where the sutdent becomes unreliable. Performance is unimodal in κ with an optimal at 1800 tokens, i.e. $0.76|\rho|_{sat}^{prop}$. Below it the summary is too coarse and the oracle intervenes more, while above it the prompt re-enters the saturation interval and validity falls. The degradation appears where the sizing rule predicts, which is a direct test of Proposition 1, rather than of the projection alone. Because K changes with κ, the experiment measures their joint effect; it does not separately identify the contribution of interval resolution or establish a flat optimum within $\pm 20\%$.

Table 26: MFBO sweep of token budget κ and interval count K. **Valid** is action validity, **Error (Point)** is decision error, **Dist. to optimal** measures the final outcome, and **Hier.** is the oracle-call rate. Both summary size and interval resolution change across rows.

κ	K	Valid ↑	Error (Point) ↓	Dist. to optimal ↓	Num. points ↑	Hier. (%)
600	4	1.00	31.4 ± 1.42	38.6 ± 1.85	138.0 ± 2.6	8.94 ± 0.42
1200	8	1.00	19.8 ± 0.94	18.2 ± 0.91	129.0 ± 2.7	5.12 ± 0.29
1800	12	0.99	14.3 ± 0.71	$\mathbf{12.74 \pm 0.62}$	121.0 ± 2.8	3.47 ± 0.21
2400	16	0.86	21.6 ± 1.08	26.9 ± 1.31	109.0 ± 3.1	6.78 ± 0.35
3000	20	0.62	44.9 ± 2.16	71.3 ± 3.02	88.00 ± 3.4	12.45 ± 0.58

Finally, in order to understand which elements are important for the projection, we next hold the protected content and budget fixed and vary how the remaining information is scored, see Table 27, in `WebArena` and `ALFWorld` case studies. This asks whether the results depend on a particular embedding or salience rule rather than on the overall projection design.

Table 27: Sensitivity to similarity and salience at fixed projection budget. MFBO reports distance to optimal; the interactive environments report success, with `ALFWorld` using the unseen split. The final row removes schema protection.

Component	Variant	**`WebArena`** ↑	**`ALFWorld`** ↑
sim	hashed bag-of-words, d=32 (deployed)	27.3	79.9
	hashed bag-of-words, d=128	27.5	80.1
	sentence-embedding cosine (MiniLM)	27.9	80.6
	exact-match indicator (no similarity)	24.1	75.2
w_g	environment salience (deployed)	27.3	79.9
	uniform, $w_g \equiv 1$	26.4	78.3
	recency, $w_g \propto$ step index	25.8	77.1
No schema protection (ablation)		19.4	68.7

Increasing the hash dimension or using learned sentence embeddings changes success only modestly in these runs. Removing coverage information or using recency alone has a larger effect, and removing schema protection produces the largest decline among the tested variants. Thus the evidence favors preserving the interface and selecting representative information, while leaving room to choose a similarity method appropriate to the available compute. These are comparisons of reported outcomes, not significance tests of the small differences between scoring methods.

A.7.5 WHEN WE NEED TO UPDATE LONG-TERM MEMORY?

The benchmark results show what the system achieves; the next experiments ask why. We begin with the output validators, because an invalid command, a stale reference, and an unproductive but admissible action require different responses. MFBO checks the response structure, fidelity range, and point domain; the browser validator in `WebArena` checks action form and, where applicable, current element references; and `ALFWorld` checks membership in the current admissible-action list. None of these checks by itself certifies task success.

Table 28: Validator-detected failure classes for distilled *Llama-3.1-8B* before projection. **Share** is the fraction of the listed failures within an environment; **Proj.** and **Upd.** are the reported fractions repaired by projection and updates at matched oracle budget. Valid but unsuccessful plans are not covered by these interface checks.

Env.	Validator	Failure class	Share	Proj.	Upd.
MFBO	regex `RESULT=[`m`,`x`]`, $m \in 1..4$, $x \in$ domain	malformed / empty	58%	0.71	0.24
		out-of-range value	42%	0.44	0.49
`WebArena`	one action from a 13-verb grammar	prose or prefix (`Action:`)	46%	0.38	0.57
		unknown verb or wrong arity	23%	0.41	0.52
		valid action, stale element id	31%	0.86	0.09
`ALFWorld`	membership in the turn's admissible-action list	not in admissible list	64%	0.79	0.17
		multi-action or commentary	36%	0.33	0.61

As seen in Table 28, projection most often repairs failures involving outdated state, such as stale browser element identifiers or actions that are no longer admissible. Parameter updates more often repair recurring response habits, such as adding prose where a command is required. This is consistent with storing episode-specific facts in the prompt and persistent corrections in the parameters. Also, we observe that it does not imply that all failures have a single cause, or that changing one mechanism can never affect another failure class. To examine when each mechanism is most useful, we consider two variations of `ALFWorld` (Table 29). First, we compare short and long tasks to test whether projection becomes more valuable when the agent must retain a longer interaction history. Second, starting at task 40, we change the object-naming convention and retain that change in subsequent tasks. This tests whether parameter updates become more valuable when a correction remains relevant across episodes, rather than describing only the current task.

Table 29: Gains in percentage points by `ALFWorld` regime. **ΔProj.** compares projection with distillation alone; **ΔUpd.** is the further contribution of updates at matched oracle budget. Short tasks use the two shortest-horizon categories; long tasks use two-object placement. The shifted stream changes object naming at task 40.

Regime	Mean horizon	ΔProj.	ΔUpd.	Dominant
Short horizon, stationary	6.7	+6.2	+7.9	updates
Full pool, stationary	9.7	+13.4	+4.5	projection
Long horizon, stationary	14.3	+21.4	+3.6	projection
Full pool, shifted at 40	9.7	+9.1	+12.8	updates

Projection contributes more on the long-horizon pool, where the student must retain more episode history. Updates contribute more when naming conventions change persistently, and also in the shortest tasks, where there is less history to compress. The comparison explains why no single allocation is uniformly preferable, indeed, the useful destination depends on whether the missing information describes the present episode or a recurring pattern. The controller uses the same observable signals across these regimes rather than receiving a regime label.

In light of the aforementioned, the trigger must make this decision without knowing the oracle's response. We assess its fidelity on a separate trace of 2000 steps per environment, labelled offline by querying the oracle at every step. This calibration expenditure is separate from deployment call rates. Precision measures how often a trigger event corresponds to oracle disagreement, and recall measures how much of that disagreement is detected. Disagreement is a proxy for possible correction, not proof that the oracle action is better.

Table 30: Trigger calibration against offline oracle-disagreement labels. Left: firing rate and disagreement prevalence (percent), precision (P), recall (R), and F1. Right: the `WebArena` uncertainty-threshold sweep; SR is success rate in percent. The selected threshold is $\tau_u = 0.45$.

Env.	Fire	Disag.	P	R	F1
MFBO	3.9	3.5	0.84	0.90	0.87
`WebArena`	6.8	6.2	0.74	0.81	0.77
`ALFWorld`	5.2	4.8	0.79	0.86	0.82

τ_u	Fire	P	R	SR
0.30	19.4	0.41	0.94	27.6
0.45	6.8	0.74	0.81	27.3
0.55	5.1	0.79	0.68	26.1
0.65	4.4	0.81	0.59	24.9
0.80	2.1	0.93	0.31	23.4

Table 30 shows how the uncertainty threshold balances disagreement detection against oracle use: a lower threshold catches more disagreements but requires more calls, while a higher threshold saves calls at the cost of missed corrections. We select a threshold beyond which additional calls yield little improvement in reported success. However, recall remains below one, meaning that the trigger still misses some opportunities for correction that may affect performance over a long episode. Oracle calls are also only part of the computational cost, since estimating uncertainty itself takes time. The default estimator uses several sampled decodings, whereas single-pass alternatives reuse logits from greedy generation. Table 31 compares their accuracy–latency trade-off at a fixed oracle budget, allowing us to assess whether cheaper uncertainty estimates preserve performance without relying on additional oracle calls.

Table 31: Uncertainty estimators on `WebArena` with *Llama-3.1-8B*. **Passes** is the reported number of decoding passes and **Latency** their added unbatched time. P/R compare against calibration labels; SR is end-to-end success. Single-pass scores are normalized and calibrated separately.

Estimator	**Passes**	**Latency** (ms)	**P**	**R**	**SR** (%)
k-sample entropy (k=4), default	4	+930	0.74	0.81	$\mathbf{27.3 \pm 3.2}$
k-sample entropy (k=2)	2	+310	0.68	0.72	26.4 ± 3.1
Mean token entropy (single pass)	1	+2	0.70	0.76	26.6 ± 3.1
Logit margin (single pass)	1	+1	0.69	0.74	26.2 ± 3.1
Free energy (single pass, calibrated)	1	+1	0.63	0.71	25.4 ± 3.1
Schema + stall + length only (no *(ii)*)	1	0	–	–	23.9 ± 3.0

It is interesting to observe that an uncertainty signal improves success relative to omitting it. Among the tested estimators, mean token entropy approaches the default's performance with much less additional computation. Batched sampling reduces the default's reported overhead, but does not eliminate it. This makes single-pass entropy a reasonable latency-oriented configuration, while the multi-sample estimator supplies the main reported results. It should not be described as identical to `PivotRL`'s pivot score, which is based on outcome variation under local rollouts.

An oracle correction may help with the current decision without warranting a parameter update, so we evaluate the update gate separately from the trigger. Over 600 deployment steps, we assess the effect of updates using a fixed set of 200 schema prompts in `WebArena` and a semantic probe. Together, these measure whether the gate preserves schema knowledge while still allowing useful semantic learning. We also include fixed-count and random-selection controls to determine whether any benefit comes from choosing better updates or simply applying fewer of them, see Table 32.

Table 32: Update-control comparison on `WebArena` with *Llama-3.1-8B*. **Probe** and **Sem.** are final schema and semantic probe accuracies. The comparison is count-matched only within the indicated groups.

Update control	**Updates**	**Probe** ↑	**Sem.** ↑	**SR** (%)	**Extra cost**
No updates (projection + correction only)	0	1.00	0.42	24.8 ± 3.1	–
Always update when fired	17	0.91	0.55	26.1 ± 3.1	–
Fixed schedule, matched count	13	0.92	0.51	25.6 ± 3.1	–
Random 13 of the fired steps	13	0.95	0.53	26.0 ± 3.1	–
EWC (Kirkpatrick et al., 2017)	17	0.97	0.54	26.8 ± 3.2	Fisher est.
OGD (Farajtabar et al., 2019)	17	0.98	0.53	26.6 ± 3.2	grad. store
O-LoRA (Wang et al., 2023)	17	0.98	0.55	26.9 ± 3.2	extra adapters
I-LoRA (Ren et al., 2024)	17	0.96	**0.57**	27.0 ± 3.2	interp. buffer
OPLoRA (Xiong and Xie, 2026)	17	0.99	0.54	27.1 ± 3.2	SVD decomp.
Risk-aware gate (ours)	13	**0.99**	0.56	27.3 ± 3.2	none
Risk-aware gate + EWC	13	**0.99**	**0.58**	$\mathbf{27.9 \pm 3.2}$	Fisher est.

It is relevant to highlight that the risk gate retains more schema accuracy than the fixed-count and random controls while continuing to improve the semantic probe. `EWC`, `OGD`, and `O-LoRA` also retain schema accuracy well, and their success rates are close to the gate's. `I-LoRA`'s interpolation buffer trades a little schema retention for the best semantic probe among the fixed-count methods, consistent with its dual-memory design favoring plasticity; `OPLoRA`'s orthogonal-complement projection retains schema essentially as well as our gate, since both constructions restrict the update away from the directions the pretrained model already relies on, though `OPLoRA` does so by an explicit SVD decomposition applied to every update rather than by deciding whether to update at all. Additionally, combining the proposed rule with `EWC` gives the highest observed success in this experiment. While the statistical superiority is not demonstrated, it suggests that deciding whether to update and controlling the update direction can complement each other.

Table 33 shows how the gate and buffer settings balance semantic learning against schema retention. A restrictive gate or a large minimum buffer allows too few updates for substantial semantic improvement, whereas a permissive gate or a small buffer permits more learning but weakens retention. The gate also depends on the fraction $q_{\mathcal{D}}/d_{\mathrm{L}}$ of adapter directions left unconstrained by the data, which enters its exposure index. We estimate this fraction from

Table 33: Sensitivity to forgetting tolerance ϵ and minimum buffer size $B_{\min}$ on `WebArena`. Each sweep fixes the other parameter at its deployed value. Probe accuracies show the balance between schema retention and semantic adaptation.

ϵ	$B_{\min}$	Updates	Probe ↑	Sem. ↑	SR (%)
0.02	32	4	1.00	0.46	25.4 ± 3.1
0.05	32	8	0.99	0.52	26.6 ± 3.1
0.10	32	13	0.99	0.56	$\mathbf{27.3 \pm 3.2}$
0.20	32	16	0.96	0.56	26.9 ± 3.2
0.50	32	17	0.92	0.55	26.2 ± 3.1
0.10	8	21	0.94	0.51	25.9 ± 3.1
0.10	64	9	0.99	0.54	26.9 ± 3.2
0.10	128	5	1.00	0.48	25.8 ± 3.1

the numerical rank of a probe Jacobian using 256 samples and a relative singular-value threshold of 10^{-3}. Since this estimate may be imperfect, Table 34 tests how the gate responds when the supplied coefficient is deliberately miscalibrated. Coefficients above one are included only to stress-test the gate; they do not represent valid dimension ratios.

Table 34: Sensitivity to the coefficient used for $q_{\mathcal{D}}/d_{\mathrm{L}}$. The reference estimate is 0.50; other rows scale it. The value 2.00 is an intentionally inflated coefficient outside the range of a literal dimension ratio. Probe is final schema accuracy.

Assumed $q_{\mathcal{D}}/d_{\mathrm{L}}$	Error	Updates	Probe ↑	SR (%)	Behaviour
0.125	4× too small	17	0.91	26.1 ± 3.1	admits all fired updates
0.25	2× too small	15	0.96	26.8 ± 3.2	
0.50	estimated	13	0.99	$\mathbf{27.3 \pm 3.2}$	
1.00	2× too large	10	0.99	27.0 ± 3.2	
2.00	4× too large	6	1.00	26.0 ± 3.1	stronger suppression

From Table 34, underestimating exposure admits more updates and weakens retention, whereas overestimating it suppresses learning. Success varies only modestly over the tested perturbations, although this does not validate the rank estimator itself. Across ten stream orders, the reference estimate has a mean of 0.50 and a standard deviation of 0.04, with at most one additional or omitted update. We fix $\lambda = 0.01$ as the optimizer weight decay. Over longer deployments, accumulated exposure can also make the gate overly restrictive. To address this, the implementation resets the update counter when accuracy on the fixed schema probe returns to its initial level, which occurred twice in the 600-step runs. This reset is a practical heuristic and is not justified by the bound.

We also inspect individual updates by evaluating a fixed unseen set before and after each update. Table 35 classifies updates by whether they change success by more than one game. These evaluations are diagnostics of the saved checkpoints; they must remain separate from the data used to choose or admit updates. The gated run contains fewer harmful updates than the ungated run, while still admitting helpful ones. This supports a retention role for the gate, although the policies follow different parameter trajectories, so the counts do not identify the counterfactual effect of every rejected update. The supplied trace associates the remaining harmful gated update with a buffer dominated by one category, suggesting that buffer diversity may matter as well as buffer size.

Table 35: Before/after update diagnostics on `ALFWorld` unseen. Helped and hurt mean a change of more than one game; neutral covers smaller changes. Net is the reported cumulative change in success, in percentage points.

Policy	Updates	Helped	Neutral	Hurt	Net
Risk-aware gate (ours)	13	9	3	1	+9.7
Always update when fired	17	10	3	4	+6.1
Fixed schedule, 13 updates	13	7	3	3	+5.4

Analogously, a similar MFBO stability experiment (Table 36) examines optimization error and schema retention, showing that the risk-aware configuration applies fewer updates and preserves more schema accuracy. Because update counts differ across policies, this experiment cannot separate the benefit of choosing better updates from that of making fewer; the matched-count experiment above addresses this distinction. Schema accuracy also captures only one aspect of forgetting: preserving the response interface does not ensure that all semantic skills are retained. These findings help explain how the controller could complement agentic post-training and memory methods. PivotRL (Yi et al., 2026) selects informative states for parameter learning, while the controller also considers whether the information should remain in the current prompt. Informative-state selection could therefore supply candidate corrections for the controller to allocate between prompts and parameters, although the methods use different supervision signals and

guarantees. Likewise, summaries produced by reflection aggregation methods such as Combee (Li et al., 2026) could feed into the bounded projection. The controller thus offers a way to coordinate existing learning and memory tools.

Table 36: MFBO stability over 600 deployment steps with *Llama-3.1-8B* and GPT-5. Probe is schema accuracy at the first and last update. Update counts differ; this is not a matched-count comparison.

Update policy	**Updates**	**Probe, first ↑**	**Probe, last ↑**	**Error (Point) ↓**	**Dist. to optimal ↓**
Fixed schedule (every 30 steps)	20	1.00	0.87	16.9 ± 0.83	19.4 ± 0.96
Drift trigger, ϵ only	17	1.00	0.94	15.1 ± 0.76	14.8 ± 0.74
Risk-aware trigger (ours)	13	1.00	**0.99**	$\mathbf{14.3 \pm 0.71}$	$\mathbf{12.9 \pm 0.64}$

A.7.6 COST, ALTERNATIVE MECHANISMS, AND LIMITS OF TRANSFER

The final experiments ask whether the observed gains remain useful after accounting for their cost and for alternative ways to improve the student. We begin with elapsed time, because fewer oracle calls do not imply that every deployment step is faster than a student-only step. Table 37 reports the supplied profiling measurements over 2000 `WebArena` steps, with the student running locally and the oracle accessed through a hosted endpoint.

Table 37: Mean per-step latency on the reported `WebArena` serving setup. Amortized costs account for call frequency; the update row uses the reported average of one update per approximately 24 steps. Totals include uncertainty estimation and between-task updates.

Component	When it runs	Cost	Amortized	GPU
Observation → units	every step	6 ms	6 ms	— (CPU)
Projection $\mathcal{P}_{\mathcal{O}}$ (lazy greedy)	every step	19 ms	19 ms	— (CPU)
Student greedy decode	every step	310 ms	310 ms	71%
Uncertainty, k=4 batched	every step	340 ms	340 ms	88%
Schema validation	every step	0.2 ms	0.2 ms	— (CPU)
Oracle call (network + generation)	6.8%	1,840 ms	125 ms	idle
LoRA update (between tasks)	1/task	8,200 ms	342 ms	96%
Total, ours			**1,142** ms	
Distill-only baseline			355 ms	
Oracle-only			1,840 ms	
Ours with single-pass uncertainty			804 ms	

The full system is slower per step than the distilled student alone, but faster than oracle-only operation in this setup. It is observed how sampling for uncertainty and applying updates account for much of the additional time; while projection itself is a small component. Hence, reusing single-pass entropy reduces latency with a modest decrease in reported success. On the other hand, updates occur after the current task's outcome is recorded, but still contribute to total throughput. Likewise, projection can overlap with other work only where the pipeline's dependencies permit it.

Next, we replace the supervising oracle while keeping the student and allocation design fixed. This tests whether the reported performance requires a particular teacher. Table 38 includes two open-weight oracles in addition to the hosted models, and reports both absolute success and the fraction of each oracle's standalone success retained by the student system.

Table 38: Oracle substitution with *Llama-3.1-8B*. Oracle-only and ours are success percentages; retained is their ratio. Hier. is the oracle-call rate for the environment in that block. The final column compares the hierarchical system with its GPT-5-supervised counterpart.

Oracle	Oracle-only	Ours	Retained	Hier. (%)	Δ vs GPT-5
`WebArena`, SR (%)					
GPT-5	51.4	27.3	0.53	6.8	–
GPT-5-nano	34.9	23.1	0.66	8.4	−4.2
Qwen2.5-72B-Instruct	30.2	21.4	0.71	9.1	−5.9
Llama-3.3-70B-Instruct	27.6	20.2	0.73	9.7	−7.1
`ALFWorld` unseen, SR (%)					
GPT-5	91.0	79.9	0.88	5.2	–
GPT-5-nano	78.4	72.6	0.93	6.7	−7.3
Qwen2.5-72B-Instruct	74.6	69.4	0.93	7.1	−10.5
Llama-3.3-70B-Instruct	71.9	67.2	0.94	7.6	−12.7

Absolute success decreases with the weaker oracles in these experiments, even though the retained fraction increases. A high retained fraction therefore does not by itself imply a more capable system as a weaker reference can be easier to approximate. The open-weight runs show that the allocation can be instantiated with those supervisors, while the headline absolute results remain specific to the selected oracle and serving configuration. Open weights improve inspectability and reproducibility of inference, but do not by themselves rule out benchmark exposure during pretraining.

We then remove oracle calls during deployment. The simplest variant retains projection and performs no updates. A second variant uses the majority action among eight student samples as supervision when the trigger fires, enabling self-consistency updates without an external model. Both start from the same offline distilled checkpoint, so "oracle-free" here refers to deployment rather than to the entire training pipeline.

As seen in Table 39, self-consistency updates retain about 90% of the full system's `WebArena` success and 92% on unseen `ALFWorld`. They show that much of the improvement can be obtained without deployment-time oracle access. The remaining gap also matters, showing that a majority of student samples can share the same error and provides no independent guarantee of correctness.

Table 39: Configurations with and without deployment-time oracle calls. All start from an offline distilled student. Self-consistency uses the majority action among $k = 8$ student samples as the update target. `ALFWorld` reports unseen success; paired call rates refer to `WebArena`/`ALFWorld`.

Configuration	Supervision	`WebArena`	`ALFWorld`	Oracle calls
Distilled student only	none (Phase 1 only)	11.6 ± 2.4	56.7 ± 8.5	0
$\mathcal{P}_{\mathcal{O}}$, no oracle	none	22.3 ± 3.0	70.1 ± 8.2	0
$\mathcal{P}_{\mathcal{O}}$ + self-consistency updates	majority of k=8 student samples	24.6 ± 3.1	73.8 ± 8.0	0
Full system (GPT-5)	oracle at 6.8%/5.2%	27.3 ± 3.2	79.9 ± 7.6	6.8%/5.2%

An alternative way to improve the interface is to constrain generation directly. We test grammar-constrained decoding (Geng et al., 2023; Willard and Louf, 2023) on both distilled and non-distilled students. The tested grammar constrains response form; it does not encode every current element identifier or the complete turn-specific admissible-action list. This distinction separates syntactic validity from whether a command is executable in the current state.

Table 40: Grammar-constrained decoding and distillation for *Llama-3.1-8B*. Parse and Exec. are the reported browser response-form and state-reference rates; task columns report `WebArena` and unseen `ALFWorld` success. Latency includes the processing assigned to each configuration.

Configuration	Parse (%)	Exec. (%)	`WebArena`	`ALFWorld`	Latency
Non-distilled	46.3	44.1	4.1	34.3	310 ms
Non-distilled + constrained decoding	**100.0**	61.8	9.7	48.6	352 ms
Distilled (Phase 1)	93.7	91.2	11.6	56.7	310 ms
Distilled + constrained decoding	**100.0**	95.4	13.1	59.8	352 ms
Ours (distilled, no constraint)	99.1	98.6	27.3	79.9	1,142 ms
Ours + constrained decoding	**100.0**	99.1	27.9	80.4	1,184 ms

Constraining syntax raises parse validity to 100%, but the tested form-only constraint can still produce a command referring to an unavailable element. Distillation improves state grounding in these runs, and combining it with constrained decoding performs better than either alone. Adding the constraint to the full system yields a smaller additional gain. This does not show that constrained decoding cannot enforce referential validity. A state-dependent constraint over the current valid actions can do so. It shows that a syntax-only constraint does not replace context management. With constrained decoding, the reported oracle-call rate falls from 6.8% to 6.1% and trigger precision rises. Fewer malformed commands reduce the need for structural corrections. A form-only constraint need not remove the validator condition entirely, however, if that validator also checks current references or action-list membership.

We extend this comparison with two mechanisms that act during generation. First, constrained decoding with recovery rejects invalid state references and retries up to four times. Second, retrieval-head reweighting, following a DySCO-style approach (Ye et al., 2026), strengthens attention to retrieved information without shortening the full prompt. We test them both on the distilled baseline and as additions to the complete system.

Table 41: Decoding-time additions for *Llama-3.1-8B*. Success is reported for `WebArena` and unseen `ALFWorld`; Tok. is the reported browser prompt length. Lat. is additional latency relative to each block starting configuration, not total latency. The upper block retains full prompts; the lower block uses projection.

Configuration	`WebArena`	`ALFWorld`	Tok.	Lat.
Distilled student	11.6 ± 2.4	56.7 ± 8.5	5.9k	—
+ constrained decoding	13.1 ± 2.5	59.8 ± 8.4	5.9k	+42 ms
+ constrained decoding, recovery (r=4)	16.8 ± 2.7	64.9 ± 8.3	5.9k	+310 ms
+ retrieval-head up-weighting (DySCO-style)	18.4 ± 2.8	66.4 ± 8.2	5.9k	+180 ms
+ both	21.2 ± 2.9	70.8 ± 8.1	5.9k	+490 ms
Ours ($\mathcal{P}_{\mathcal{O}}$ + trigger + updates)	27.3 ± 3.2	79.9 ± 7.6	2.2k	—
+ constrained decoding, recovery	28.4 ± 3.2	81.0 ± 7.5	2.2k	+310 ms
+ retrieval-head up-weighting	27.9 ± 3.2	80.3 ± 7.5	2.2k	+180 ms
+ both	$\mathbf{29.1 \pm 3.3}$	$\mathbf{81.6 \pm 7.5}$	2.2k	+490 ms

As seen in Table 41, both mechanisms improve the unprojected student, and combining them narrows the gap to the full system. They also improve the full system when added on top of it, which supports their use as complementary components. The table does not constitute a globally equal-compute comparison. A deployment choice should therefore compare total latency and supervision cost alongside the additional success, rather than treating the added-latency column as the entire cost.

The remaining experiments test the scope of the attention explanation. Theorem 1 requires a suitable bound on score spread as the prefix grows. Measuring finite scores at one length is not enough to establish a length-uniform bound. We therefore inspect head-level behavior as well as the aggregate attention slope.

Table 42: Head-level attention behavior for *Llama-3.1-8B*. Score spreads and relevant-block mass are reported at length 2048; slopes use the length sweep. Peaked heads retain a top attention weight above 0.5 at every measured length.

Head group	**Share**	**median $b - a$**	**mass on R**	**fitted slope**
Diffuse (top-1 < 0.2)	61%	6.4	0.0139	-1.02
Intermediate	33%	9.1	0.0204	-0.96
Peaked / retrieval-like	6%	17.8	0.0731	-0.31
All heads (reported in Table 8)	100%	7.9	0.0176	-0.94

As shown in Table 42, most measured heads dilute at a rate close to the aggregate trend, while a small peaked group retains relevant information more strongly. This heterogeneity limits how literally an average slope should be applied to individual heads. It also explains why attention mass and action validity are different measurements, some heads can retain information while the complete model still produces invalid responses. The experiment supports attention dilution as one mechanism, without establishing that it is the sole cause of behavioral saturation.

We next ask whether an architecture designed to retain longer history removes the need for prompt control. The compressive-memory student uses an Infini-style design (Munkhdalai et al., 2024), combining local attention with a bounded memory. This is a change to the internal processing of context, rather than an online write of the current episode into the model's parameters.

Table 43: Observed saturation and controller gains for different student designs. The nominal window describes the model-supported history, while the observed threshold comes from a behavioral sweep. ΔOurs compares the full system with distillation alone for that student on `WebArena`.

Student	**Nominal window**	$\lvert\rho\rvert^{\mathbf{obs}}_{\mathbf{sat}}$	$\boldsymbol{\Delta_\rho}$	**ΔOurs**
Llama-3.1-8B (dense)	128k	2160	0.044	+15.7
Mistral-7B (dense, `SWA`)	32k	940	0.015	+12.4
Compressive-memory 8B (Infini-style)	∞	7420	0.151	+9.8

The compressive-memory model reaches its observed threshold later and gains less from the controller than the dense *Llama-3.1-8B* student, see Table 43. This is consistent with the proposed role of projection which states that better internal history management leaves less for external context management to repair. The comparison is indicative because the models are not matched in every respect. Behavioral calibration can be repeated for the new architecture, but an attention-specific formula need not transfer unchanged.

Finally, we broaden the architectural comparison to a positional-encoding ablation and models with recurrent or state-space sequence processing. Table 44 applies the same behavioral sweep while reporting an attention slope only where that measurement is defined.

Table 44: Behavioral thresholds across sequence-processing mechanisms. Slope is the measured relevant-block attention exponent where attention exists; n/a indicates that the model has no corresponding attention matrix. The `NoPE` row is an ablation probe, not a separately trained matched model.

Model	**Sequence mixing**	$\lvert\rho\rvert^{\mathbf{obs}}_{\mathbf{sat}}$	$\boldsymbol{\Delta_\rho}$	**Slope**
Llama-3.1-8B	softmax attention, `RoPE`	2160	0.044	-0.94
Mistral-7B	softmax attention, `RoPE` + `SWA`	940	0.015	-0.97
Llama-3.1-8B, `NoPE` probe	softmax attention, no positional encoding	1780	0.036	-0.91
Compressive-memory 8B	local attention + linear memory	7420	0.151	-0.62
`Mamba-2` 7B	selective state space	3980	0.081	n/a
`RWKV-6` 7B	linear recurrence	3110	0.063	n/a

Saturation persists in the `NoPE` probe, but removing positional encoding also changes a trained model, so the result does not isolate every positional effect. The recurrent and compressive models have later behavioral thresholds in these runs, yet still become unreliable as the supplied history grows. The softmax-attention theorem does not apply to `Mamba` or `RWKV`. What can transfer is the empirical procedure of measuring a reliable operating range and testing a bounded representation within it.

Taken together, the experiments distinguish the main design choice from its implementation details. Reliable deployment requires preserving the current interface, retaining enough episode state for the next decision, and deciding whether a correction is persistent enough to justify learning. Projection, oracle intervention, and parameter updates address different parts of that problem. Their relative value depends on the student, the environment, and the available computation; the ablations identify these dependencies rather than claiming that one allocation or one numerical threshold is universal.

B Experimental Setting and Reproducibility

This section summarizes the implementation choices and evaluation protocol used throughout the experiments. We consider three environments with different sources of information growth: Multi-Fidelity Bayesian Optimization (MFBO), where a numerical optimization history grows over time; `WebArena`, where large accessibility-tree observations accumulate during browser interaction; and `ALFWorld`, where the agent must retain progress over long embodied trajectories. In all cases, the same oracle–student–controller structure is used, while the state representation, output schema, and projection units are adapted to the environment.

URSA implementation. All experiments are implemented in `URSA` (Universal Research and Scientific Agent), which provides the execution layer used to instantiate the oracle, adaptive student, controller, and environment interfaces. Agents follow the common `BaseAgent` abstraction and implement their local model or tool together with a `forward()` method. A minimal agent follows the form

```
from ursa.agents.base_agent import BaseAgent

class AdaptiveAgent(BaseAgent):
    def forward(self, state):
        return self.model.generate(state)
```

The framework can be installed as

```
git clone https://github.com/lanl/ursa.git
cd ursa
pip install -e .
```

Models and architecture. We instantiate the adaptive student $\mathcal{U}_\theta$ using *Llama-3.1-8B* and *Mistral-7B*. The main oracle models $\mathcal{O}_\phi$ are *GPT-5* and *GPT-5-nano*, accessed through hosted API endpoints. Oracle corrections during deployment are generated at temperature 0. To test whether the allocation mechanism depends on a particular closed model, additional experiments replace the hosted oracle with the open-weight models *Qwen2.5-72B-Instruct* and *Llama-3.3-70B-Instruct*.

At deployment step t, the controller receives the accumulated state S_t, preserves the protected schema information, and projects its semantic component into a bounded state $\widetilde{\mathsf{S}}_t$. The student then produces

$$y_t = \mathcal{U}_\theta(\widetilde{\mathsf{S}}_t).$$

The response is checked by the environment-specific schema validator and by the controller signals used to detect semantic or interaction drift. Importantly, an oracle response is not required to decide whether the oracle should be called.

Practical implementation of oracle supervision. The distillation objective in the main text is expressed through a KL divergence between the oracle and student policies. The hosted oracle models, however, are treated as black-box teachers and do not expose their complete vocabulary distributions. We therefore do not numerically evaluate the KL divergence. Instead, for an oracle completion $y^\star = (y_1^\star, \dots, y_L^\star)$, the student minimizes the teacher-forced negative log-likelihood

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{\ell=1}^{L} w_\ell \log \pi_\theta \left(y_\ell^\star \mid y_{<\ell}^\star, \widetilde{\mathsf{S}}_t \right).$$

Averaging this objective over sampled oracle completions gives a Monte Carlo estimator of the cross-entropy term in the KL objective; the omitted oracle-entropy term does not depend on θ and therefore does not affect optimization.

For schema-related corrections, tokens defining the interaction protocol receive weight $w_{\mathrm{protocol}} = 4.0$, while the remaining tokens receive unit weight. For semantic corrections, the complete oracle response is used without protocol-specific reweighting. Thus, oracle logits are never required for student training.

Phase 1: offline distillation. Before deployment, each environment supplies 5000 structured examples supervised by *GPT-5*. The purpose of this phase is to provide the student with the response interface required by the environment before online adaptation begins. The construction of the training data is separated from evaluation differently in each environment. MFBO uses synthetic states generated with seeds disjoint from the evaluation seeds. `WebArena` uses

agent rollouts on the self-hosted environments with separate training and evaluation task templates. `ALFWorld` uses its 3553 training games and excludes all 140 seen and 134 unseen evaluation games.

Learning the output interface may also transfer semantic information and is therefore not interpreted as syntax-only training. The Phase 1 sweeps in the main appendix explicitly vary the number of examples and epochs to quantify this transfer.

MFBO environment. In MFBO, the agent maximizes an unknown objective

$$x^\star = \arg\max_{x \in \mathcal{X}} F(x)$$

using four fidelity models with execution costs of 1, 2, 3, and 4 minutes. At each iteration, the prompt contains the available fidelity models, previous evaluation points, observed errors, active domain, uncertainty, and remaining time. The required output protocol is

```
RESULT=[model,point]
```

so that the agent must jointly select a discrete fidelity and a continuous evaluation point under a wall-clock budget. The fixed response grammar and fidelity information form the protected schema component, while the growing optimization history forms the semantic state.

`WebArena` environment. `WebArena` contains 812 tasks on self-hosted shopping, administration, discussion, software-development, and map sites. At every interaction, the student receives the task goal, previous actions, and the current accessibility tree. The output must consist of exactly one of the 13 admissible browser actions and no additional explanation. The controller therefore protects the task goal and action grammar while treating accessibility-tree nodes and previous actions as selectable semantic information.

`ALFWorld` environment. `ALFWorld` contains 140 seen and 134 unseen evaluation games across six household-task categories, with episodes lasting at most 40 steps. Unlike `WebArena`, the admissible-action set changes at every step. The controller therefore protects the current task, observation, and admissible actions, while previous trajectory information is treated as the compressible semantic state. This state contains information such as explored locations, object positions, container states, inventory, completed subgoals, and previous actions.

Prompt projection. The projection operator is implemented through the weighted facility-location objective introduced in the main text. Given the decomposition

$$\mathsf{S}_t = \mathcal{Z}_{\text{sch}} \oplus \mathcal{Z}_{\text{sem}},$$

the controller first reserves space for all protected units $\mathcal{Z}_{\text{sch}}$. The remaining semantic units compete for the residual token budget according to marginal coverage gain per token. If the protected component alone exceeds the available budget, the state is declared non-deployable rather than silently removing protocol information.

The semantic unit depends on the environment. In MFBO, one unit is an evaluation tuple (x_i, e_i, I_i, u_i) and similarity is computed from min–max normalized numerical fields. In `WebArena`, units are accessibility-tree nodes or previous actions and are represented using 32-dimensional hashed bag-of-words signatures. Interactive nodes receive salience 2.0, previous actions 1.5, and remaining nodes 1.0. In `ALFWorld`, one unit is an $(\text{action}, \text{observation})$ trajectory step, again represented by a 32-dimensional hashed signature; steps that change the observation receive salience 1.5 and the remaining steps weight 1.0.

For the main *Llama-3.1-8B* experiments, the configured token budgets are

$$\kappa_{\text{MFBO}} = 1800, \qquad \kappa_{\text{WebArena}} = 2200, \qquad \kappa_{\text{ALFWorld}} = 1400.$$

These values are deployment budgets and should not be confused with the observed saturation thresholds measured independently in the prompt-length experiments.

Controller and oracle trigger. The deployed controller relies only on quantities available from the student or environment. A correction is considered when the response violates the output schema, the environment fails to change for $n = 5$ steps, or student uncertainty exceeds

$$\tau_u = 0.45.$$

The default uncertainty estimator uses $k = 4$ sampled student decodings at temperature 0.7. For discrete actions, uncertainty is measured through action entropy, while the continuous MFBO decision uses a coefficient of variation. The threshold τ_u is calibrated on a held-out trace. In particular, oracle disagreement is not an input to the deployed trigger; otherwise, determining whether to query the oracle would itself require querying it at every step.

Online parameter adaptation. Oracle corrections are stored in a bounded buffer $\mathcal{B}$. Parameter adaptation can occur only once

$$|\mathcal{B}| \geq B_{\min} = 32,$$

and the buffer contains at most 256 examples. Before applying an update, the controller evaluates the catastrophic-forgetting proxy introduced in the main text, using forgetting tolerance $\epsilon = 0.10$ and exposure coefficient 0.50. The resulting gate determines whether the buffered information is consolidated into the student parameters or retained for a later update.

Student adaptation uses AdamW with learning rate

$$\eta = 2 \times 10^{-4}$$

and weight decay $\lambda = 0.01$. LoRA adapters use rank 32, scale 64, and dropout 0.05, and are applied to the attention and MLP projections,

```
q_proj, k_proj, v_proj, o_proj,
gate_proj, up_proj, down_proj.
```

This results in approximately 90.18M trainable parameters for *Llama-3.1-8B* and 86.80M for *Mistral-7B*.

Avoiding within-task information leakage. All configurations are evaluated on the same ordered task stream. During an individual task, projection and immediate oracle correction may affect subsequent actions, but the student parameters remain fixed. Oracle corrections are buffered and parameter updates are applied only after the current task terminates. Consequently, the parameters used at the beginning of task τ can contain supervision from previous tasks, but never from task τ itself. In `ALFWorld`, the adapted student persists across tasks within a split and is reset between the seen and unseen evaluation splits.

Metrics and statistical evaluation. For all environments, we report schema validity as the fraction of generated outputs accepted by the corresponding protocol validator. Semantic drift is measured using an ℓ_1 error for numerical decisions and cosine-based comparison for semantic decisions when appropriate.

MFBO reports the final distance to the optimum, number of evaluations, protocol violations, oracle usage, and point- and fidelity-selection errors. Confidence intervals for the end-to-end MFBO comparisons are obtained through paired bootstrap resampling over 20 matched seeds.

`WebArena` reports task success, action validity, average prompt length, oracle-call rate, and external oracle cost over all 812 tasks. `ALFWorld` reports success separately on the 140 seen and 134 unseen games, together with solved-episode length, inadmissible-action rate, looping frequency, and oracle-call rate. Binomial success rates are reported with Wilson 95% confidence intervals. Since all methods operate on the same task sets, paired comparisons use McNemar tests on discordant outcomes, with Holm correction applied within each benchmark. The seen and unseen `ALFWorld` splits are pooled only for the reported paired hypothesis tests.

Matched comparisons. Where oracle supervision is compared across methods, we explicitly match the oracle-call budget so that performance differences are not explained only by additional teacher access. Similarly, continuous-learning comparisons use the same ordered task stream. The frozen-adapter control is first adapted on a disjoint stream of 200 tasks and is subsequently evaluated with its parameters fixed while retaining projection and oracle correction. The online-adaptation configuration instead allows admitted parameter updates between evaluation tasks.

Compute. Student fine-tuning is performed locally using parameter-efficient LoRA updates on a single NVIDIA Tesla V100 16GB GPU. The oracle models are accessed separately through hosted API calls and are not fine-tuned. Reported oracle-call rates and external API expenditures are therefore separated from local student-training cost. The open-weight-oracle experiments replace the hosted teacher while leaving the student, controller, projection, and evaluation protocols unchanged.