ARES: Adaptive Reasoning Effort Selection for Efficient LLM Agents

Jingbo Yang[1*] **Bairu Hou**[1] **Jiayun Wang**[2] **Wei Wei**[2] **Yujia Bao**[2†] **Shiyu Chang**[1†]

[1]Department of Computer Science, University of California, Santa Barbara
[2]Center for Advanced AI, Accenture

Abstract

Modern agents powered by thinking LLMs achieve high accuracy through long chain-of-thought reasoning but incur substantial inference costs. While many LLMs now support configurable reasoning levels (*e.g.*, high/medium/low), static strategies are often ineffective: using low-effort modes at every step leads to significant performance degradation, while random selection fails to preserve accuracy or provide meaningful cost reduction. However, agents should reserve high reasoning effort for difficult steps like navigating complex website structures, while using lower-effort modes for simpler steps like opening a target URL. In this paper, we propose ARES, a framework for per-step dynamic reasoning effort selection tailored for multi-step agent tasks. ARES employs a lightweight router to predict the lowest appropriate reasoning level for each step based on the interaction history. To train this router, we develop a data generation pipeline that identifies the minimum reasoning effort required for successful step completion. We then train the router to predict these levels, enabling plug-and-play integration for any LLM agents. We evaluate ARES on a diverse set of agent tasks, including `TAU-Bench` for tool use agents, `BrowseComp-Plus` for deep-research agents, and `WebArena` for web agents. Experimental results show that ARES reduces reasoning token usage by up to 52.7% compared to fixed high-effort reasoning, while introducing minimal degradation in task success rates. Code is available at `https://github.com/UCSB-NLP-Chang/Ares`.

1 Introduction

Recent advancements in reasoning large language models (LLMs) [Team et al., 2025, Guo et al., 2025, Comanici et al., 2025] have significantly boosted the capabilities of autonomous agents [Jin et al., 2025, Sun et al., 2025, Liu et al., 2025b, Yang et al., 2025c] in complex, multi-step decision-making tasks. By leveraging extended chain-of-thought (CoT) reasoning, these agents can perform deeper environment analysis and more rigorous planning before executing actions. However, this improved performance comes at a substantial inference cost, as a large number of reasoning tokens are accumulated at each step along the multi-step trajectories. To control these costs, a straightforward approach is to leverage the configurable "thinking levels" (or reasoning efforts) now supported by various state-of-the-art LLMs Singh et al. [2025]. These models (*e.g.*, the `GPT-5` or `Gemini-3`) allow users to manually select from thinking modes, such as *high/medium/low* or *thinking/fast* modes, to balance performance and budget. With these options, users can configure LLM agents to always reason at lower levels at each step to reduce the cost.

However, such a uniform approach is often suboptimal because not all steps in a task require the same level of thinking. While some steps are simple, others demand intensive reasoning to avoid

*Correspondence to: <jingbo@ucsb.edu>
†Equal advising.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

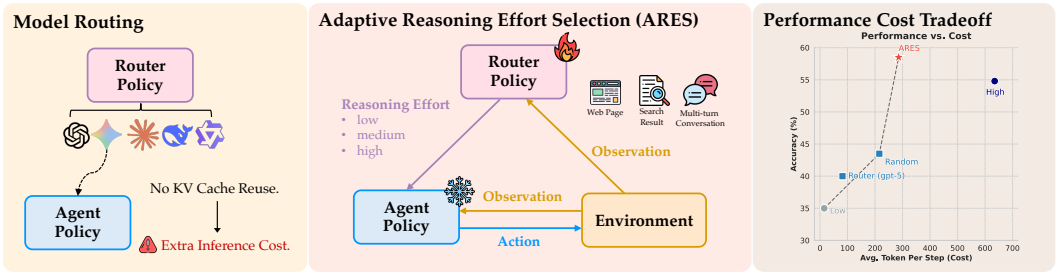

Figure 1: **Overview of the Adaptive Reasoning Effort Selection (ARES) Framework. Left:** Traditional Model Routing which often incurs extra inference costs without KV cache reuse. **Middle:** Our proposed ARES framework, dynamically allocates reasoning effort at each step. **Right:** ARES (red star) achieves the optimal balance between performance and cost compared to baselines.

errors. Consequently, a fixed, static strategy often fails to balance performance and cost effectively. For example, a naive approach that consistently applies a low reasoning effort to minimize costs leads to severe performance degradation. As illustrated in Figure 1, the agent powered by `gpt-oss-20b` suffers a nearly 20% drop after switching the reasoning effort from "high" to "low" at every decision step. This performance gap suggests that a coarse-grained reduction in thinking effort is insufficient for maintaining agent effectiveness in complex environments.

To address these challenges, we propose a framework for dynamic reasoning effort allocation tailored for LLM agents. The core idea is to move beyond static configurations by adaptively determining the most suited reasoning effort for each individual step, reducing reasoning costs while preserving performance. While adaptive thinking has been explored in single-step tasks such as mathematical reasoning and competitive programming Shen et al. [2025], Cui et al. [2025], its application to LLM agents remains non-trivial. A slight reasoning deficit in an early step can lead to error propagation, making the balance between efficiency and long-term success far more complex. Furthermore, our approach distinguishes itself from existing model routing strategies Su et al. [2025], Zhang et al. [2025a], Amayuelas et al. [2025], where user inputs and tasks are routed to models with different sizes and capabilities, yielding a performance-cost trade-off that is neither predictable nor monotonic. In contrast, by leveraging the intra-model thinking levels (*e.g.*, switching between a model's own high and low reasoning modes), we ensure a well-defined and consistent performance-cost frontier with finer-grained control of the agent and no integration overhead from maintaining multiple heterogeneous models. The intra-model paradigm additionally lets the agent reuse the KV cache across reasoning-effort switches (Appendix K), avoiding the re-encoding latency that traditional model routing incurs and maximizing the token-saving benefits of our adaptive framework.

Under this setting, we propose ARES, a method that introduces a small-scale LM as a reasoning-effort router to work alongside the LLM agent. Specifically, the router takes the current interaction history as input and directly predicts the lowest appropriate reasoning-effort level for the next step. The agent then uses the predicted reasoning level to perform the next step. This design is model-agnostic and can be seamlessly integrated into any existing agent architecture. We train this lightweight router (e.g., `Qwen3-1.7B`) by standard supervised fine-tuning (SFT) as well as reinforcement learning (RL). During SFT stage, we develop a multi-phase automated data generation pipeline that identifies and labels the minimum reasoning effort required for each step within a trajectory. Specifically, we break down the multi-step reasoning effort labeling task as a combination of single-turn classification problems. We start with collecting high-quality trajectory data to obtain the ground truth agent actions. After that, we annotate the lowest reasoning effort which produces the correct action stably, and train the router to predict these effort levels by minimizing the next-token prediction loss. Subsequently, we employ RL to optimize the router's strategy. By interacting with the environment, the router learns to maximize a combined reward to balance both task performance as well as reasoning cost.

We validate ARES through comprehensive experiments across diverse agent tasks, including tool-use agents, deep-research agents, and web agents. Specifically, we fine-tune a lightweight `Qwen3-1.7B` model as the reasoning-effort router in ARES. Compared to consistently using high reasoning effort, ARES substantially reduces token usage while maintaining task performance. For example, on `TAU-Bench`, ARES achieves an 52.7% reduction in reasoning cost and even slightly improves performance relative to always using high reasoning effort.

2 Related Work

In this section, we provide an overview of prior works which closely relate to or partially motivated our work, including LLM routing, efficient reasoning.

LLM Routing. Routing across models of varying scales or architectures is a well-established technique for balancing performance and cost. Early methods [Srivatsa et al., 2024, Ong et al., 2024, Feng et al., 2024] trained routers—often using human preference data [Chiang et al., 2024]—to select the optimal model per task. Other approaches employ clustering-based routing, such as Avengers [Zhang et al., 2025c], BESTRoute [Ding et al., 2025], and related frameworks [Jitkrittum et al., 2025]. However, these primarily target single-turn tasks (*e.g.*, math), reducing routing to an independent classification problem. In contrast, ARES addresses multi-turn decision-making, where the effort chosen at one step shapes the states the agent faces at later steps; its RL stage learns this cross-step allocation from trajectory-level outcomes.

Recent research has extended model routing to multi-turn agent tasks. ToolOrchestra [Su et al., 2025] utilizes an orchestrator LLM for joint planning and routing, though it relies on subjective natural language descriptions of model capabilities. Router-R1 [Zhang et al., 2025a] frames multi-turn routing as a sequential process but focuses on simple QA tasks (*e.g.*, NQ [Kwiatkowski et al., 2019], TriviaQA [Joshi et al., 2017]) that often collapse into single-turn RAG problems. More recently, EvoRoute [Zhang et al., 2026] proposed an experience-driven self-routing framework. While these multi-model paradigms show promise, they remain constrained by non-monotonic cost-performance relationships and inefficiencies from redundant context encoding, as detailed in Section 1. In contrast, our setting is a better defined optimization problem, and reuses the KV cache to avoid additional inference cost.

Efficient and adaptive LLM reasoning. Our method is also related to token reduction in generative models [Kong et al., 2025] and, more specifically, to recent approaches for adaptive thinking of LLMs, which aim to teach models to reason adaptively according to the input difficulty [Liu et al., 2025a, Yu et al., 2025, Wu et al., 2025, Zhang et al., 2025b] or a user-specified thinking budget [Huang et al., 2025, Alomrani et al., 2025, Hou et al., 2025]. DynaThink [Pan et al., 2024] selects a fast or slow reasoning strategy once per query, and Think Smarter not Harder [Yu et al., 2025] fine-tunes the reasoning model with inference-budget-constrained policy optimization on single-turn mathematical problems. Similar to other works on model routing, these approaches primarily focus on controlling reasoning in a single-turn setting, by dynamically adjusting the length of reasoning traces or truncating intermediate thoughts under a given budget. In contrast, our approach models reasoning effort as a sequential decision-making process, making it applicable to more complex, multi-turn agent tasks.

3 Method

In this section, we present ARES, a framework that dynamically allocates most efficient but effective reasoning effort across decision steps. We first provide the problem formulation of optimizing router model, and then elaborate how we train an effective reasoning effort router using both supervised fine-tuning (SFT) as well as reinforcement learning (RL).

3.1 Problem Formulation

To formalize the decision-making process of the router, we first define the agent's task and interaction environment. An agent task is characterized by a goal or user query $x \in \mathcal{X}$. The interaction proceeds over discrete turns $t = 1, \ldots, T$. At each turn t, the environment provides an observation $o_t \in \mathcal{O}$ (*e.g.*, external tool outputs or web page content). The LLM agent $\mathcal{M}_{\text{agent}}$ with parameter ϕ first performs chain-of-thought reasoning and then predicts the next action a_t. Notably, the reasoning level e_t of the LLM is configurable and can be selected from a fixed set, such as high/medium/low supported by `gpt-oss-20b`. Let $h_t = (x, o_1, a_1, \ldots, o_{t-1}, a_{t-1})$ denote the interaction history up to turn t, the agent model $\mathcal{M}_{\text{agent}}$ with parameters ϕ predicts the next action a_t:

$$a_t \sim P_\phi(a_t \mid h_t, o_t, e_t) \tag{1}$$

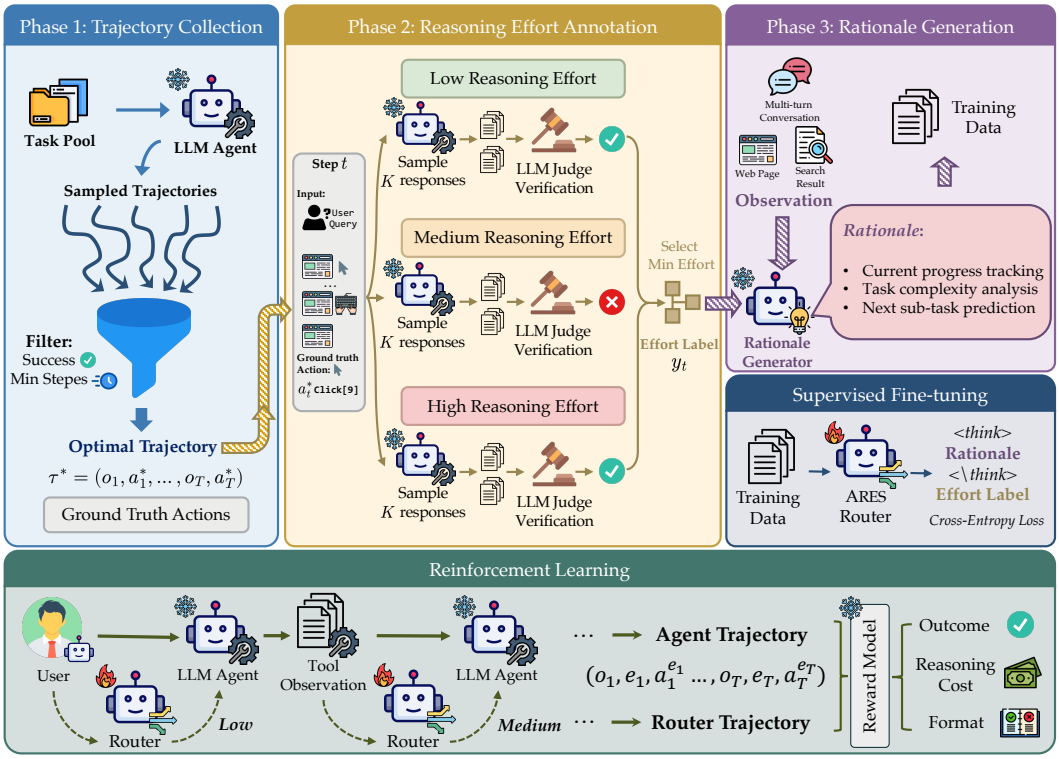

Figure 2: **Overview of the ARES training pipeline.** (1) Trajectory Collection: Optimal ground-truth paths are defined by filtering successful trajectories with minimal steps. (2) Effort Annotation: The minimum sufficient reasoning effort for each step is identified via sampling and LLM verification. (3) Rationale Generation: A teacher LLM generates semantic justifications based on task observations and complexity. (4) Supervised Fine-tuning: The ARES router is fine-tuned to jointly predict rationales and effort labels. (5) Reinforcement Learning: The fine-tuned ARES router will be further trained using GRPO with outcome, reasoning cost, and format reward.

The agent's objective is to produce a sequence of actions such that the complete trajectory $\tau = (x, o_1, a_1, \ldots, o_T, a_T)$ satisfies the task success criterion, denoted by a verification function $\mathcal{V}(\tau) \in \{0, 1\}$.

The router is a lightweight LLM $\mathcal{M}_{\text{router}}$ with parameters θ. At each turn t, the router receives the same input context as the agent—the task, history, and current observation—and predicts the optimal reasoning effort level e_t from a discrete space $\mathcal{E} = \{e_{\text{low}}, e_{\text{mid}}, e_{\text{high}}\}$: $e_t = \mathcal{M}_{\text{router}}(h_t, o_t; \theta)$. The selected e_t represents the reasoning strategy that $\mathcal{M}_{\text{agent}}$ will employ for that specific turn.

The objective of the router is to minimize the cumulative inference cost across the trajectory while ensuring that the task remains successful. We quantify the cost of each step, $\text{cost}(e_t)$, as the total number of tokens generated by the agent at turn t, which includes both the internal reasoning (thinking) tokens and the tokens for the final action. Formally, the optimization goal is to find a selection policy that maximizes the task success rate while minimizing the cumulative computational cost:

$$\max_{\theta} \mathbb{E}_{x \sim \mathcal{X}, \tau \sim \mathcal{T}(\theta, \phi)} \left[\mathcal{V}(\tau, x) - \lambda \sum_{t=1}^{T} \text{cost}(e_t) \right] \tag{2}$$

where $\tau \sim \mathcal{T}(\theta, \phi)$ is the trajectory distribution on task x induced by the LLM agent parametrized by ϕ and the router parameterized by θ. $\mathcal{V}(\tau, x)$ measures whether the task success of trajectory τ on task x. By solving this optimization problem, the router learns to dynamically allocate minimal reasoning resources at each step while preserving the final task utility.

3.2 Supervised Fine-tuning Pipeline

The optimal effort level at any step is not directly observable, and brute-force search over $|\mathcal{E}|^T$ joint configurations is intractable. Worse, a sub-optimal effort at an early step propagates errors to later observations and inflates the cost of every downstream step, coupling all decisions together. We sidestep this by *decoupling* effort selection from trajectory search: anchor on a high-quality successful trajectory, then label each step independently against the corresponding ground-truth action.

Phase 1: Trajectory collection. For each task x we sample N successful trajectories under the maximum effort e_{high} and retain the *shortest* one, $\tau^* = (o_1, a_1^*, \ldots, o_T, a_T^*)$, as the reference path. Keeping the shortest trajectory cleans the data: longer successful trajectories often contain loops, failed actions, and redundant recovery steps that would otherwise be labeled. The pipeline is also compatible with publicly released agent trajectory datasets, allowing ARES to leverage existing high-quality interactions without always relying on self-sampling.

Phase 2: Effort annotation. Treating each a_t^* as ground truth, we identify the lowest effort level that reliably reproduces it. To absorb LLM stochasticity, for every $e \in \mathcal{E}$ we draw K independent samples $\{\hat{a}_{t,k}^{(e)}\}_{k=1}^{K}$ and accept e as *sufficient* if it matches in at least M of K trials under a domain-specific action-equivalence rule (*e.g.*, identical tool name and key arguments for tool-use, identical element ID for web agents, semantically equivalent query for deep research; LLM-judge reliability is verified in Appendix I). The candidate set of sufficient levels is

$$\mathcal{C}_t = \Big\{ e \in \mathcal{E} \;\Big|\; \sum_{k=1}^{K} \mathcal{V}\big(\hat{a}_{t,k}^{(e)}, a_t^*\big) \geq M \Big\}, \tag{3}$$

and we assign the label $y_t = \arg\min_{e \in \mathcal{C}_t} c(e)$. When $\mathcal{C}_t = \emptyset$, we fall back to the lowest effort with the highest non-zero match rate, and discard the step only if no sample at any effort level matches a_t^* (Appendix J.3).

Phase 3: Rationale generation. A powerful teacher LLM produces a short rationale r_t (3–5 sentences) explaining why y_t is appropriate, integrating the complexity of the current observation, the progress of the overall task, and the estimated difficulty of the next sub-task. The rationale serves as a chain-of-thought supervision target that helps the lightweight router analyze the state explicitly rather than mapping context directly to a label, and we ablate its contribution in Section 4.6.

Supervised fine-tuning. The router is fine-tuned on $\mathcal{D} = \{(h_t, o_t, r_t, y_t)\}$ with a standard next-token prediction loss, jointly emitting the rationale and the effort label. This step-wise objective is a tractable surrogate for the trajectory-level cost in Section 3.1; the full equivalence rules, teacher prompting, and the conditions under which the surrogate tightly bounds the trajectory optimum are deferred to Appendix J.

3.3 Reinforcement Learning

While SFT trains the ARES router to maximize the probability of selecting the minimum sufficient reasoning effort at each step, the SFT data synthesis pipeline reduces the multi-turn routing process to a series of single-step decisions with a greedy objective. This approach may fail to capture the complex multi-turn dynamics of the routing process. Specifically, the greedy nature of SFT introduces two primary shortcomings: ❶ single-step selection assumes that all previous effort levels were optimal, leaving the router without training signals on how to recover from sub-optimal selections made in prior turns; ❷ the current SFT data provides only one step-wise optimal selection per query, lacking the contrastive signals necessary to optimize the total query-wise reasoning effort across different selection sequences. To address these challenges and fully realize the potential of the ARES router, we further explore training via reinforcement learning (RL).

Router Rollout. Following the formulation in Section 3.1, at each agent decision turn t, the ARES router takes the trajectory history h_t and the current observation o_t as input to produce: (1) a rationale r_t (chain-of-thought), where the router analyzes the current task progress and the difficulty of the next sub-task; and (2) a reasoning effort prediction e_t. This selected effort level is then provided as part of

the input to the downstream agent to generate the subsequent action $a_t^{e_t}$, as is shown in Figure 2. The rollout continues until the task is successfully completed or the agent reaches the maximum number of allowed steps.

Reward Design. We balance task success against computational efficiency through a composite reward $R(\tau)$ on a trajectory τ of length T, optimized with Group Relative Policy Optimization (GRPO) [Shao et al., 2024]:

$$R(\tau) = \begin{cases} R_{\text{out}} + R_{\text{cost}} + R_{\text{form}} & \text{if the task succeeds} \\ R_{\text{out}} + R_{\text{form}} & \text{otherwise.} \end{cases} \tag{4}$$

The *outcome* reward $R_{\text{out}} = +5.0$ on success and 0 otherwise dominates the signal so the router prioritizes task completion. The *cost* reward penalizes effort with $c(e_t) = -0.2, -0.5, -1.0$ for $e_{\text{low}}, e_{\text{mid}}, e_{\text{high}}$ and is normalized to the trajectory-average $R_{\text{cost}} = \frac{1}{T}\sum_{t=1}^{T} c(e_t)$; we apply R_{cost} *only* when the trajectory succeeds, otherwise the router learns a degenerate policy that picks low effort to fail quickly and minimize accumulated penalties. The *format* reward $R_{\text{form}} = -1.0$ fires when the prescribed `<think>`-tagged response template is violated; the rollout is then terminated and marked as failed, since we empirically found that routing through a malformed turn destabilizes training. This structure ensures the dominant correctness signal R_{out} takes priority, with R_{cost} shaping the efficiency–accuracy trade-off only after correctness is achieved.

RL Data Filtering. Training signal quality is paramount, so we curate the prompt pool with a two-stage filter using $N = 8$ rollouts per prompt from the SFT-initialized router: (i) we discard prompts with zero success across all rollouts, since such tasks exceed the agent's capability and the router would be incorrectly punished for inherent agent failures; (ii) among prompts where the agent always succeeds, we retain the top 30% by reward variance, where multiple effort sequences yield the same outcome at very different costs and therefore provide the strongest signal for learning efficient routing. Full filtering rationale and implementation details are in Appendix C.5.

4 Experiments

4.1 Experimental Setup

Environments and backbone. We evaluate ARES on three diverse agent benchmarks: `TAU-Bench` [Yao et al., 2024] for agent tool use, `BrowseComp-Plus` [Chen et al., 2025, Wei et al., 2025] for deep-research agents over a fixed retrieval corpus, and `WebArena` [Zhou et al., 2023] for web navigation with accessibility-tree observations. The default agent backbone is `gpt-oss-20b`; we also evaluate ARES on `Qwen3-8B/32B` [Yang et al., 2025a] (Appendix H).

Training data. Training trajectories are sourced from `APIGen-MT` [Prabhakar et al., 2025] for tool use, rejection sampling within our own pipeline (Section 3.2) for deep research, and the successful trajectories released by AGENTOCCAM [Yang et al., 2024] for web navigation.

Evaluation. We report task success rate for `WebArena`, average reward for `TAU-Bench`, and accuracy (judged by `GPT-4o`, following the official implementation; Appendix I) for `BrowseComp-Plus`. Reasoning cost is measured by three metrics: total tokens consumed across the entire benchmark (T_{total}), average tokens per task (T_{task}), and average tokens per agent step (T_{step}).

Baselines. We compare against five baseline families: (i) *fixed-effort* policies that always select low, medium, or high reasoning effort, approximating the lower and upper bounds of performance; (ii) a *random* policy that uniformly samples among the three effort levels at each step; (iii) a training-free *prompting-based* router that calls a large reasoning LLM (*e.g.*, `GPT-5`, `Gemini-3-Pro`) to select effort; (iv) *learned routers* with released checkpoints (TCAndon-Router [Zhao et al., 2026], Router-R1 [Zhang et al., 2025a], and ToolOrchestra [Su et al., 2025]; Section 4.4); and (v) confidence-based adaptive-reasoning methods (DEER [Yang et al., 2025b] and Think-or-Not [Yong et al., 2025]; Appendix D). End-to-end cost and latency, including every router call, are reported in Section 4.3 and Appendix G. The agent frameworks, annotation pipeline configuration, and full SFT/RL hyperparameters are in Appendix C.

Table 1: Main evaluation results in **TAU-Bench**, **BrowserComp-Plus**, and **WebArena**, using `gpt-oss-20b` as the backbone LLM. Arrows (↑ / ↓) indicating an increase or decrease in absolute value. Orange and blue indicate beneficial and detrimental changes. †Accuracies taken directly from the `gpt-oss` model card [Agarwal et al., 2025].

	Method	Performance			Efficiency (Token Consumption)					
		Acc. (%)	Δ_{Acc}	S	$T_{\textbf{total}}$	Δ_{token}	$T_{\textbf{task}}$	Δ_{token}	$T_{\textbf{step}}$	Δ_{token}
TAU-Bench *Retail*										
RULE -BASED	Low	35.0†	↑ 19.8	13.5	25k	↑ 627k	223	↑ 5454	15	↑ 373
	Medium	47.3†	↑ 7.5	14.5	137k	↑ 515k	1198	↑ 4479	82	↑ 306
	High	54.8†	↓ 0.0	13.8	1007k	↓ 355k	8756	↓ 3079	634	↓ 246
	Random	43.5	↑ 11.3	14.5	359k	↑ 293k	3126	↑ 2550	215	↑ 172
PROMPTING -BASED	GPT 5	40.0	↑ 14.8	14.0	132k	↑ 520k	1148	↑ 4529	81	↑ 307
	Gemini 3 Pro	46.1	↑ 8.7	14.7	128k	↑ 524k	1115	↑ 4562	76	↑ 312
	ARES	54.8	–	14.6	652k	–	5677	–	388	–
TAU-Bench *Airline*										
RULE -BASED	Low	32.0†	↑ 4.0	10.8	12k	↑ 666k	250	↑ 13327	23	↑ 1077
	Medium	42.6†	↓ 6.6	13.2	98k	↑ 580k	1961	↑ 11616	148	↑ 952
	High	38.0†	↓ 2.0	10.6	873k	↓ 195k	17472	↓ 3895	1654	↓ 554
	Random	34.0	↑ 2.0	13.5	521k	↑ 157k	10437	↑ 3140	770	↑ 330
PROMPTING -BASED	GPT 5	34.0	↑ 2.0	12.4	396k	↑ 282k	7927	↑ 5650	641	↑ 459
	Gemini 3 Pro	36.0	↑ 0.0	12.6	239k	↑ 439k	4779	↑ 8798	379	↑ 721
	ARES	36.0	–	12.3	678k	–	13577	–	1100	–
BrowseComp-Plus										
RULE -BASED	Low	8.00	↑ 33.3	4.6	5k	↑ 1066k	332	↑ 6449	72	↑ 113
	Medium	34.0	↑ 7.3	26.2	538k	↑ 533k	3590	↑ 3191	137	↑ 48
	High	42.7	↓ 1.4	55.4	1841k	↓ 770k	12276	↓ 5495	220	↓ 35
	Random	30.7	↑ 10.6	18.0	392k	↑ 679k	2616	↑ 4165	145	↑ 40
PROMPTING -BASED	GPT 5	38.7	↑ 2.6	45.1	1398k	↓ 327k	9321	↓ 2540	206	↓ 21
	Gemini 3 Pro	37.3	↑ 4.0	41.6	1144k	↓ 73k	7628	↓ 847	183	↑ 2
	ARES	41.3	–	36.5	1071k	–	6781	–	185	–
WebArena										
RULE -BASED	Low	37.4	↑ 9.1	8.9	67k	↑ 1445k	520	↑ 11203	58	↑ 1266
	Medium	42.6	↑ 3.9	9.9	538k	↑ 974k	4170	↑ 7553	420	↑ 904
	High	45.0	↑ 1.5	10.0	2763k	↓ 1251k	21424	↓ 9701	2154	↓ 830
	Random	40.3	↑ 6.2	8.0	857k	↑ 655	6643	↑ 5080	830	↑ 494
PROMPTING -BASED	GPT 5	41.1	↑ 5.4	9.0	1159k	↑ 353k	8990	↑ 2733	1001	↑ 323
	Gemini 3 Pro	41.9	↑ 4.6	9.1	1164k	↑ 348k	9023	↑ 2670	995	↑ 329
	ARES	46.5	–	8.9	1512k	–	11723	–	1324	–

4.2 SFT Results

Table 1 shows that the SFT-trained ARES router consistently matches or exceeds the fixed high-effort baseline while substantially reducing computational overhead. Per-benchmark accuracy is 54.8% on `TAU-Bench` *Retail* (tied with the high baseline), 36.0% on *Airline*, 41.3% on `BrowseComp-Plus` (within 1.4pp of the 42.7% high-effort ceiling), and 46.5% on `WebArena`, which actually exceeds the 45.0% high-effort baseline. The total reasoning-token reductions (T_{total}) are approximately 35.2% on *Retail*, 41.8% on `BrowseComp-Plus`, and 45.3% on `WebArena` — i.e., ARES delivers high-effort-level performance at a fraction of the cost.

Beyond these aggregate numbers, three domain-specific phenomena highlight when dynamic effort allocation is most useful. *(i) Overthinking on* Airline *and* `WebArena`. On `TAU-Bench` *Airline* we observe a non-monotonic relationship where the medium-effort baseline surpasses the high-effort one — excessive reasoning leads to logic drift. The same effect surfaces on `WebArena`, where high effort produces divergent reasoning that manifests as malformed actions or lost task focus. By dynamically modulating effort, ARES avoids the overthinking, outperforming the high-effort baseline on `WebArena`. *(ii) Long-horizon sensitivity on* `BrowseComp-Plus`. The large gap between low (8.0%) and high (42.7%) baselines shows that deep-research tasks are extremely sensitive to per-step effort: a single under-thought retrieval step yields a poorly formulated query, which cascades into missing information for the remainder of the task. ARES's near-ceiling accuracy at substantial

Table 2: RL results in **TAU-Bench** using `gpt-oss-20b` as the backbone LLM. Arrows (↑ / ↓) indicating an increase or decrease in absolute value. Orange and blue indicate beneficial and detrimental changes.

	Method	Performance			Efficiency (Token Consumption)					
		Acc. (%)	Δ_{Acc}	S	$T_{\textbf{total}}$	Δ_{token}	$T_{\textbf{task}}$	Δ_{token}	$T_{\textbf{step}}$	Δ_{token}
TAU-Bench *Retail*										
RULE -BASED	Low	35.0	↑ 23.5	13.5	25k	↑ 451k	223	↑ 3918	15	↑ 271
	Medium	47.3	↑ 11.2	14.5	137k	↑ 339k	1198	↑ 2943	82	↑ 204
	High	54.8	↑ 3.7	13.8	1007k	↓ 531k	8756	↓ 4615	634	↓ 348
	Random	43.5	↑ 15.0	14.5	359k	↑ 117k	3126	↑ 1015	215	↑ 71
ARES	SFT	54.8	↑ 3.7	14.6	652k	↓ 176k	5677	↓ 1536	388	↓ 102
	RL	58.5	–	14.4	476k	–	4141	–	286	–
TAU-Bench *Airline*										
RULE -BASED	Low	32.0	↑ 10.0	10.8	12k	↑ 121k	250	↑ 2403	23	↑ 208
	Medium	42.6	↓ 0.6	13.2	98k	↑ 35k	1961	↑ 692	148	↑ 83
	High	38.0	↑ 4.0	10.6	873k	↓ 740k	17472	↓ 14819	1654	↓ 1423
	Random	34.0	↑ 8.0	13.5	521k	↓ 388k	10437	↓ 7784	770	↓ 539
ARES	SFT	36.0	↑ 6.0	12.3	678k	↓ 545k	13577	↓ 10924	1100	↓ 869
	RL	42.0	–	11.5	133k	–	2653	–	231	–

token savings indicates that it accurately pinpoints the small set of critical reasoning nodes that cannot be down-routed. *(iii) Routing overhead vs. proprietary routers.* Across all benchmarks ARES incurs far lower routing overhead than the prompting-based `GPT-5` and `Gemini-3-Pro` routers; on *Retail* the gap is also accuracy-wise — ARES outperforms the GPT-5 router by 14.8% — on *Airline* ARES reaches parity with these much larger routers, and on `WebArena` the substantial success-rate gain over their static configurations demonstrates a stronger ability to handle dynamic web environments at a fraction of the routing cost.

4.3 RL Results

To evaluate the impact of our reinforcement learning phase, we compare the performance and efficiency of ARES trained solely with SFT against ARES further optimized with RL. We conduct this evaluation on both the `TAU-Bench` Retail and Airline domains.

As shown in Table 2, the RL optimization consistently improves both task accuracy and token efficiency. In the *Retail* domain, RL training increases the success rate from 54.8% to 58.5% while simultaneously reducing total token consumption by 176k compared to the SFT baseline. This dual improvement is even more pronounced in the *Airline* domain, where the RL-optimized router achieves a 6.0% absolute increase in accuracy and drastically cuts total token usage by nearly 80%.

Furthermore, compared to the static high effort rule-based strategy, ARES (RL) achieves superior accuracy across both domains while consuming less than half the total reasoning tokens (476k vs. 1007k in Retail; 133k vs. 873k in Airline). These findings confirm that the RL phase effectively pushes the Pareto frontier, enabling the router to discover optimal, context-aware reasoning effort selections that the greedy SFT objective fails to capture.

End-to-end cost and latency. The token savings carry over to wall-clock time and monetary cost once every router call is counted (Appendix G). With the router served on one H100 and the agent on two H100s, ARES completes *Retail* in 2438.3 s versus 3524.1 s for the high-effort baseline, a 30.8% reduction, and the router itself adds only 8.5% wall-clock overhead. At public OpenRouter pricing, the total cost of a *Retail* run drops from \$0.2014 to \$0.0966 (52.0%), of which the router accounts for \$0.0019.

To further understand the RL optimization, we analyze the training dynamics in the `TAU-Bench` Airline domain, where the agent exhibits a counter-intuitive "overthinking" phenomenon: the high reasoning effort yields a lower accuracy (38.0%) than the medium setting (42.6%). Figure 3(a) illustrates how GRPO training effectively corrects this bias. While the SFT-initialized router initially selects high effort for over 50% of the steps, the RL process rapidly suppresses this detrimental over-deliberation, dropping its usage to under 20%. Simultaneously, the router learns to confidently

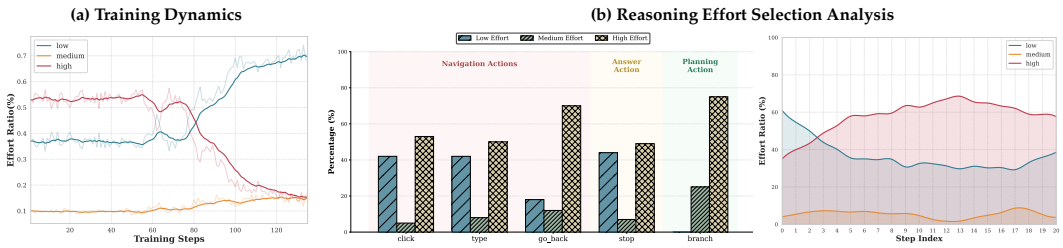

Figure 3: **Reasoning-effort dynamics and analysis. (a)** Evolution of ARES effort selection during GRPO training on `TAU-Bench` *Airline*. **(b)** Distribution of low/medium/high effort across task step indices on `WebArena`. Distribution of effort levels grouped by action type on `WebArena`.

Table 3: Comparison with learned routers on **TAU-Bench**, using their released checkpoints. *Effort* routes among the low/medium/high modes of `gpt-oss-20b`; *Model* routes among `Qwen3-4B`, `gpt-oss-20b`, and `Qwen3-32B`. T_{task}: reasoning tokens per task. O_{task}: output tokens per task converted into `gpt-oss-20b`-equivalent tokens by each selected model's output price.

			Retail			Airline		
Method	**Space**	**Router**	**Acc. (%)**	T_{task}	O_{task}	**Acc. (%)**	T_{task}	O_{task}
TCAndon	Effort	4.0B	39.1	3152	4240	42.0	6015	8129
Router-R1	Effort	3.1B	41.7	2310	3370	42.0	3868	5155
ToolOrchestra	Effort	8.2B	41.7	1521	2572	38.0	3544	5348
TCAndon	Model	4.0B	50.4	2481	26221	34.0	3350	31882
Router-R1	Model	3.1B	45.2	2307	25485	38.0	3032	28788
ToolOrchestra	Model	8.2B	47.0	2116	25394	40.0	2726	29347
ARES	Effort	1.7B	58.5	4141	6039	42.0	2653	3681

rely on more efficient strategies, with the low effort ratio climbing to approximately 70%. By autonomously curbing unnecessary computation, the RL-optimized ARES avoids the overthinking trap and maximizes both task success and token efficiency.

4.4 Comparison with Learned Routers

We compare ARES against three learned routers with released checkpoints: TCAndon-Router [Zhao et al., 2026], Router-R1 [Zhang et al., 2025a], and ToolOrchestra [Su et al., 2025]. Each router is evaluated in two routing spaces: *effort routing* over the low/medium/high modes of `gpt-oss-20b`, and *model routing* over `Qwen3-4B`, `gpt-oss-20b`, and `Qwen3-32B`. Besides reasoning tokens per task (T_{task}), we report all output tokens per task converted into `gpt-oss-20b`-equivalent tokens by each selected model's output price (O_{task}), which makes mixed-model trajectories comparable. The protocol and the routers' own overhead are detailed in Appendix E.

As shown in Table 3, ARES reaches 58.5% on *Retail*, 8.1 points above the strongest learned router (TCAndon with model routing, 50.4%). Model routing spends over $4\times$ more equivalent output tokens than ARES (25.4k–26.2k vs. 6.0k), while effort routing with the learned routers spends fewer tokens but trails ARES by at least 16.8 points. On *Airline*, ARES ties the best accuracy (42.0%) and uses the fewest reasoning and equivalent output tokens of all methods.

4.5 Analysis of Reasoning Effort Selection

To better understand the decision-making logic of the router, we analyze the distribution of reasoning effort along two dimensions on `WebArena`: temporal progression and action categories. For our AGENTOCCAM agent, which has no separate planning phase, Figure 3(b) shows that in the early stages of a task (*e.g.*, steps 0–2) the router predominantly selects low reasoning effort, because initial navigation steps often involve low-complexity environments such as a forum or landing page, where the mapping from observation to action is relatively straightforward. As the task progresses, the frequency of high reasoning effort increases significantly, driven by the escalating perceptual

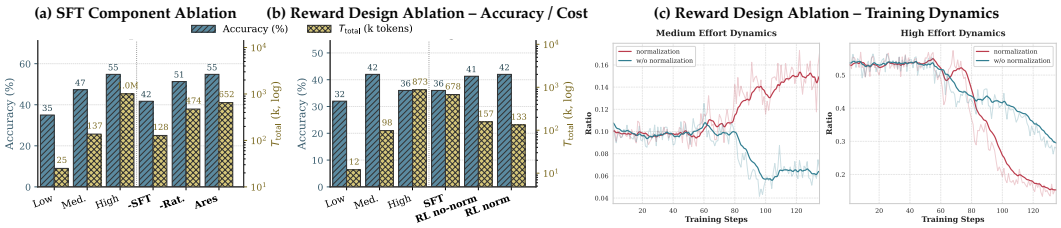

Figure 4: Ablation studies of ARES on TAU-Bench. **(a)** SFT component ablation (Retail) and **(b)** reward-design ablation (Airline): striped bars report accuracy (%), cross-hatched bars report total reasoning tokens (T_{total}, log scale). **(c)** GRPO training dynamics (Airline): medium- and high-effort selection ratios with vs. without the normalized cost reward.

complexity of the web observations and the expanding context window. This temporal pattern follows the states the agent encounters rather than the step index: a router that samples effort from the same step-conditioned distribution loses 7.0 points of accuracy (Appendix F).

Figure 3(b) further breaks down effort by action type. The critical decision steps, specifically `go_back` and `branch`, require the highest proportion of high reasoning effort. In our framework, a `go_back` action typically signifies a strategic recovery from an incorrect navigation path, while a `branch` action indicates a substantive modification of the existing navigation plan. The concentration of intensive reasoning at these actions indicates that ARES identifies them as high-stakes error correction or plan refinement steps, where high effort is essential for sophisticated self-correction mechanisms.

4.6 Ablation Studies

We conduct ablation studies on the effects of generated rationale, SFT, and reward design in ARES.

Effect of Supervised Fine-tuning. As shown in the "- SFT" setting in Figure 4(a), removing fine-tuning and using `Qwen3-1.7B` out-of-the-box leads to the most significant degradation, with accuracy dropping from 54.8% to 41.7%. Although this setting has the lowest token consumption, the low accuracy indicates that without task-specific tuning, the base model lacks the judgment to allocate sufficient reasoning effort, often defaulting to overly simplistic strategies.

Effect of Reasoning Rationale. We evaluate Phase 3 by training a router that predicts effort labels directly without rationales ("- Rationale"), resulting in a 3.5% accuracy drop. This confirms that, by forcing the router to explicitly analyze task difficulty before deciding, the internal thinking process acts as a necessary cognitive bridge that significantly improves the accuracy of effort selection.

Reward Design. We compare the trajectory-averaged cost reward used by ARES against an accumulated variant $R_{\text{cost}} = \sum_{t=1}^{T} c(e_t)$ in which we proportionally scale the per-step penalties to $-0.02, -0.06, -0.12$ for low/medium/high effort so that successful trajectories still yield a net positive reward. Figure 4(b,c) shows that normalization is essential: it drives the high-effort selection ratio down to $\sim$15% (vs. $\sim$30% without normalization), steadily increases reliance on the domain-optimal medium effort, and yields both higher task accuracy and $\sim$15% fewer total tokens. A step-by-step walk-through of the GRPO training dynamics behind these numbers is provided in Appendix M.

5 Conclusion

We introduced ARES, a framework that optimizes LLM agent efficiency by dynamically selecting appropriate reasoning effort levels based on task complexity. Experimental results on enriched benchmarks like `BrowseComp-Plus` and `TAU-Bench` demonstrate that ARES achieves performance parity with high-effort strategies while significantly reducing reasoning token consumption. Furthermore, our evaluation confirms that the learned reasoning patterns generalize effectively across different model scales, model thinking configuration, and data sources.

Acknowledgments and Disclosure of Funding

The UCSB team acknowledges support from the National Science Foundation (NSF) under Grants IIS-2338252, IIS-2302730, and BCS-2619240. The authors declare no competing interests.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, et al. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*, 2025.

Alfonso Amayuelas, Jingbo Yang, Saaket Agashe, Ashwin Nagarajan, Antonis Antoniades, Xin Eric Wang, and William Wang. Self-resource allocation in multi-agent llm systems. *arXiv preprint arXiv:2504.02051*, 2025.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Yingqian Cui, Zhenwei Dai, Pengfei He, Bing He, Hui Liu, Xianfeng Tang, Jingying Zeng, Suhang Wang, Yue Xing, Jiliang Tang, et al. Adaptive test-time reasoning via reward-guided dual-phase search. *arXiv preprint arXiv:2509.25420*, 2025.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks VS Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute. *arXiv preprint arXiv:2506.22716*, 2025.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, et al. Universal model routing for efficient llm inference. *arXiv preprint arXiv:2502.08773*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Zhenglun Kong, Yize Li, Fanhu Zeng, Lei Xin, Shvat Messica, Xue Lin, Pu Zhao, Manolis Kellis, Hao Tang, and Marinka Zitnik. Token reduction should go beyond efficiency in generative models–from vision, language to multimodality. *arXiv preprint arXiv:2505.18227*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. Diffadapt: Difficulty-adaptive reasoning for token-efficient llm inference. *arXiv preprint arXiv:2510.19669*, 2025a.

Yujian Liu, Ze Wang, Hao Chen, Ximeng Sun, Xiaodong Yu, Jialian Wu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Shiyu Chang. Learning from online videos at inference time for computer-use agents. *arXiv preprint arXiv:2511.04137*, 2025b.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. DynaThink: Fast or slow? A dynamic decision-making framework for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14686–14695, 2024.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*, 2025.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from llm routing. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 124–134, 2024.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, et al. Toolorchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Chao Wu, Baoheng Li, Mingchen Gao, and Zhenyi Wang. From efficiency to adaptivity: A deeper look at adaptive reasoning in large language models. *arXiv preprint arXiv:2511.10788*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025b.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for llm-based web agents. *arXiv preprint arXiv:2410.13825*, 2024.

Yuhao Yang, Zhen Yang, Zi-Yi Dou, Anh Nguyen, Keen You, Omar Attia, Andrew Szot, Michael Feng, Ram Ramrakhya, Alexander Toshev, et al. Ultracua: A foundation model for computer use agents with hybrid action. *arXiv preprint arXiv:2510.17790*, 2025c.

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

Xixian Yong, Xiao Zhou, Yingying Zhang, Jinlin Li, Yefeng Zheng, and Xian Wu. Think or not? exploring thinking efficiency in large reasoning models via an information-theoretic lens. *arXiv preprint arXiv:2505.18237*, 2025.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, et al. Think smarter not harder: Adaptive reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*, 2025.

Guibin Zhang, Haiyang Yu, Kaiming Yang, Bingli Wu, Fei Huang, Yongbin Li, and Shuicheng Yan. Evoroute: Experience-driven self-routing llm agent systems. *arXiv preprint arXiv:2601.02695*, 2026.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Ruofan Zhang, Bin Xia, Zhen Cheng, Cairen Jian, Minglun Yang, Ngai Wong, and Yuan Cheng. Dart: Difficulty-adaptive reasoning truncation for efficient large language models. *arXiv preprint arXiv:2511.01170*, 2025b.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, et al. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants. *arXiv preprint arXiv:2505.19797*, 2025c.

Jiuzhou Zhao, Chunrong Chen, Chenqi Qiao, Lebin Zheng, Minqi Han, Yanchi Liu, Yongzhou Xu, Xiaochuan Xu, and Min Zhang. TCAndon-Router: Adaptive reasoning router for multi-agent collaboration. *arXiv preprint arXiv:2601.04544*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Prompts

In this section, we display the prompts used for the router, reasoning effort annotator, and rationale generator in ARES. We include the prompts used in `WebArena` with the `gpt-oss-20b` backbone as an example.

Router

```
You are a router that selects the reasoning_effort (low, medium, or
high) for the OSS model's next response.
Context: The input is the current query of an OSS-model-based web
agent completing the objective, including the objective, web page
observation, previous interaction history, etc. The OSS model needs
to decide the action for the next step and eventually complete the
objective. The OSS model incurs higher cost and latency at higher
reasoning levels.
- Choose HIGH if the next step likely needs careful reasoning, or
complex tool sequencing.
- Choose MEDIUM if the next step is moderately complex.
- Choose LOW if the next step is straightforward and the risk is low.
Produce exactly one word: low, medium, or high.
```

Reasoning Effort Annotator

```
You are a strict evaluator of web navigation actions.
Two actions match if and only if they are exactly identical after
trimming leading and trailing whitespace and collapsing consecutive
whitespace characters into a single space.
You must enforce exact equivalence. Do NOT allow any differences in:
- action type or name,
- arguments or argument values,
- URLs, selectors, or identifiers.
If any difference exists, the actions must be judged as NOT matching.
```

Rationale Generator

```
You are annotating router training data for a web-navigation agent.
Background: The router selects the reasoning_effort (low, medium,
high) for an OSS model's next action. Higher effort means more
careful reasoning and higher cost/latency. The context includes the
user's objective and the agent's interaction history leading up to
the next action.
Task: Given the context, the chosen label, as well as OSS model
response from different efforts, write up to five concise sentences
explaining why this label is appropriate. Ground your explanation in
the objective and interaction history in the given context. Reveal
chain-of-thought or step-by-step reasoning for why you choose the
corresponding reasoning effort through analyzing current task states
and predicting the difficulty of next sub-task.
```

B Training Example

In this section, we provide a training example used for SFT.

Training Example

```
System Prompt: You are a router that selects the reasoning_effort
(low, medium, or high) for the OSS model's next action.
Context: One web agent interacting with web elements to complete
objectives. The OSS model charges more cost and latency at higher
reasoning levels.
- Choose HIGH if the next step likely needs careful reasoning, or the
current observation contains complex patterns.
- Choose MEDIUM if the next step is moderately complex.
- Choose LOW if the next step is straightforward and the risk is low.
Produce exactly one word: low, medium, or high.
User Prompt: OBJECTIVE: Where is the nearest In-N-Out to Upitts, and
what is the walking distance to it?
INTERACTION HISTORY: Step 0: REASON: To proceed with finding the
nearest In-N-Out to Upitts, the first logical step is to use the
search function provided by OpenStreetMap. By entering the query for
the nearest In-N-Out to Upitts in the search textbox and submitting
it, we can potentially gather information on the nearest location.
This will help in identifying the location before proceeding to
calculate the walking distance. ACTION: type [35] [In-N-Out
near University of Pittsburgh] [1] OBSERVATION: RootWebArea [1]
'OpenStreetMap' heading 'OpenStreetMap logo OpenStreetMap' link [41]
'OpenStreetMap logo OpenStreetMap' link [402] 'Edit' button [403]
link [373] 'History' link [374] 'Export' link [407] 'GPS Traces' link
[408] 'User Diaries' link [409] 'Communities' link [410] 'Copyright'
link [411] 'Help' link [412] 'About' link [382] 'Log In' link [383]
'Sign Up' link [515] 'Where is this?' textbox [35] 'Search' button
[516] 'Go' link [503] 'Find directions between two points' heading
'Welcome to OpenStreetMap!' button [469] 'Close' text 'OpenStreetMap
is a map of the world, created by people like you and free to use
under an open license.'
Label: High
Rationale: The objective requires finding the nearest In-N-Out
relative to the University of Pittsburgh and computing a walking
distance, which likely involves interpreting search results,
verifying the correct location, and using the directions tool. The
current page exposes multiple relevant actions, such as search and
route planning, that require careful selection. Since In-N-Out
locations are uncommon near Pittsburgh, identifying the correct
nearest location is nontrivial and demands careful reasoning.
Therefore, a high reasoning effort is appropriate.
```

C Implementation Details

This appendix collects the full implementation, data, and training details that are summarized in Section 4.

C.1 Environment Details

TAU-Bench [Yao et al., 2024] evaluates agent tool-use within task-oriented dialogues across *retail* and *airline* domains. Each sample involves an interaction between a user simulator and an LLM agent, where the agent must execute database tool calls to resolve user queries or complete tasks (*e.g.*, flight bookings). Success is measured by the correctness of the final database state or the provided information. We invoke ARES after every user message and tool observation.

BrowseComp-Plus [Chen et al., 2025], a controlled derivative of BrowseComp [Wei et al., 2025], ensures experimental reproducibility by replacing open-web search with a fixed retrieval corpus

Table 4: Dataset statistics of reasoning-effort labels across benchmarks.

Dataset	Total	High	Medium	Low
TAU-Bench	43,358	12,261	6,222	24,875
BrowseComp-Plus	12,366	6,184	3,091	3,091
WebArena	1,718	1,095	72	551

and incorporating human verification. The benchmark evaluates agents in a multi-step setting that interleaves reasoning and retrieval: at each step the agent generates a reasoning sequence followed by a search action or final answer. ARES is invoked prior to each reasoning and action step.

`WebArena` [Zhou et al., 2023] provides functional websites across domains such as e-commerce, social forums, and software development. Agents receive webpage observations (*e.g.*, accessibility trees) and perform human-like actions (*e.g.*, click, scroll, type) to fulfill queries that require information seeking, navigation, or configuration. ARES is invoked before every agent inference step.

C.2 Agent Backbones and Frameworks

The default agent backbone for all main-paper results is `gpt-oss-20b`. For `TAU-Bench` we follow the original implementation and use `GPT-4o` as the user-simulator model. For the deep-research agent we use `BM25` as the retriever and reuse the native web-search and browsing tools shipped with the `gpt-oss` family. For web agents we adopt AGENTOCCAM [Yang et al., 2024] as the underlying agent framework. Beyond `gpt-oss-20b`, we also evaluate the ARES pipeline on `Qwen3-8B/32B` (with the model's native *think*/*no-think* hybrid modes); the corresponding results are reported in Appendix H.

C.3 Training Data and Dataset Statistics

Initial training trajectories combine publicly released agent trajectories with trajectories collected by our own pipeline (Section 3.2). For tool-use training we use `APIGen-MT` [Prabhakar et al., 2025], which provides conversational trajectories with ground-truth actions at every step. For `BrowseComp-Plus` we collect trajectories using rejection sampling within our pipeline. For `WebArena` we use the successful trajectories released by AGENTOCCAM. For both `BrowseComp-Plus` and `WebArena` we adopt the train-test splits used by prior work [Sun et al., 2025, Qi et al., 2024, Yang et al., 2024].

Table 4 reports the resulting distribution of reasoning-effort labels per benchmark after running the data-construction pipeline of Section 3.2.

C.4 Annotation Pipeline Configuration

For the robustness check during reasoning-effort annotation we sample $K = 3$ trials per effort level and use `GPT-4o` as the LLM judge. We accept an effort level only if it produces the correct action in all three trials ($M = K = 3$). For rationale generation we use `GPT-5` as the teacher model. If no effort level passes the three-out-of-three check at a step, we fall back to the lowest effort level with the highest non-zero pass rate; the step is discarded only if no trial at any effort level produces the correct action (Appendix J.3).

C.5 RL Prompt Filtering

This subsection expands the two-stage prompt filter summarized in Section 3.2. For every prompt in the candidate pool we perform $N = 8$ rollouts using the SFT-initialized router and use the resulting per-rollout success indicators and total rewards to filter the prompts.

Stage 1 — Zero-Success Filtering. We compute the success rate (SR) for each prompt across the N rollouts and discard any prompt with SR=0. Such prompts likely exceed the fundamental capabilities of the underlying agent LLM regardless of the reasoning effort applied. Including these

"unsolvable" samples would introduce significant noise into the training signal, since the router could be incorrectly punished for the agent's inherent failures rather than for poor effort allocation.

Stage 2 — Variance-Based Efficiency Selection. For the remaining prompts on which the agent achieves SR=100%, we compute the variance of the total reward across the N rollouts and retain only the top 30% of prompts ranked by this reward variance. High reward variance on a fully solved prompt indicates that different effort sequences produce the same successful outcome at very different costs — exactly the situation where the router has the largest opportunity to reduce reasoning tokens without sacrificing accuracy. Conversely, prompts with low reward variance among successful rollouts contribute weak gradient signal because the choice of effort barely changes the cost.

By curating the data in this manner, the RL stage focuses on prompts where reasoning-effort selection is the primary driver of the efficiency–accuracy trade-off.

C.6 Training Hyperparameters

Supervised fine-tuning. We fine-tune the ARES router for 3 epochs with a learning rate of 5×10^{-6}, a global batch size of 64, a warmup ratio of 0.1, and a weight decay of 0.01.

Reinforcement learning. The RL stage is initialized from the SFT checkpoint. For each prompt we generate $G = 16$ rollouts to compute the GRPO group-wise advantage, and cap the prompt and response lengths at 4,096 and 512 tokens, respectively. We use the Adam optimizer with a constant learning rate of 1.5×10^{-6} and a KL-divergence coefficient of 0.01 against the reference model. Training spans 5 epochs with a global batch size of 32.

C.7 Evaluation Protocol Clarifications

The three reasoning-cost metrics reported in the main paper form a hierarchy:

- T_{step} — average reasoning tokens per single agent step. The input context at any step strictly respects the `gpt-oss-20b` 131K context window.
- T_{task} — average total reasoning tokens per task trajectory; this aggregates T_{step} across the multiple steps required to complete one task.
- T_{total} — total reasoning tokens consumed when evaluating the full benchmark dataset; this aggregates T_{task} across all evaluated tasks and naturally exceeds the 131K single-inference limit.

D Comparison with Adaptive Reasoning Baselines

Beyond the rule-based and prompting-based baselines reported in the main paper, we further compare ARES against two representative families of *intra-model* adaptive reasoning methods that do not rely on a learned router. Both families have been studied extensively in the single-turn reasoning literature.

Confidence-based truncation. Following Dynamic Early Exit in Reasoning Models (DEER) [Yang et al., 2025b], at every reasoning "transition point" (*e.g.*, the words `wait`, `alternatively`, etc.), we temporarily stop the chain-of-thought and force the agent to commit to a candidate next action. If the model's confidence on this action exceeds a threshold δ, the reasoning is terminated early; otherwise, the agent continues thinking. We sweep $\delta \in \{0.50, 0.75, 0.95\}$.

Entropy-based gating. Following the entropy-based adaptive reasoning method Think-or-Not [Yong et al., 2025], we compute the entropy of the agent's distribution over the valid tool-call action space (*e.g.*, the entropy over tool names such as `search_direct_flight` and `update_reservation_flights` on `TAU-Bench` *Airline*). The entropy is computed over tool-call *names* only and excludes their parameters. Whenever the entropy falls below a threshold η, we stop the reasoning and immediately sample the final action. We sweep $\eta \in \{0.1, 0.2, 0.3\}$.

Table 5 reports the results across both `TAU-Bench` domains. ARES (RL) consistently outperforms every threshold setting of both baselines on accuracy. While the threshold-based baselines can reduce

Table 5: Comparison with confidence-based and entropy-based adaptive reasoning baselines on **TAU-Bench**, using `gpt-oss-20b` as the backbone LLM. We sweep multiple thresholds for each baseline. Arrows (↑ / ↓) indicate the change relative to ARES (RL). Orange and blue indicate beneficial and detrimental changes.

	Method	Performance		Efficiency (Token Consumption)					
		Acc. (%)	Δ_{Acc}	$T_{\textbf{total}}$	Δ_{token}	$T_{\textbf{task}}$	Δ_{token}	$T_{\textbf{step}}$	Δ_{token}
TAU-Bench *Retail*									
CONFIDENCE -BASED	DEER ($\delta = 0.50$)	52.2	↑ 6.3	171k	↑ 305k	1486	↑ 2655	100	↑ 186
	DEER ($\delta = 0.75$)	53.9	↑ 4.6	401k	↑ 75k	3489	↑ 652	237	↑ 49
	DEER ($\delta = 0.95$)	56.5	↑ 2.0	463k	↑ 13k	4030	↑ 111	278	↑ 8
ENTROPY -BASED	Think-or-Not ($\eta = 0.1$)	51.3	↑ 7.2	293k	↑ 183k	2544	↑ 1597	166	↑ 120
	Think-or-Not ($\eta = 0.2$)	53.0	↑ 5.5	275k	↑ 201k	2390	↑ 1751	153	↑ 133
	Think-or-Not ($\eta = 0.3$)	53.9	↑ 4.6	262k	↑ 214k	2281	↑ 1860	146	↑ 140
	ARES (RL)	58.5	–	476k	–	4141	–	286	–
TAU-Bench *Airline*									
CONFIDENCE -BASED	DEER ($\delta = 0.50$)	30.0	↑ 12.0	114k	↑ 19k	2276	↑ 377	162	↑ 69
	DEER ($\delta = 0.75$)	36.0	↑ 6.0	236k	↓ 103k	4729	↓ 2076	356	↓ 125
	DEER ($\delta = 0.95$)	36.0	↑ 6.0	321k	↓ 188k	6421	↓ 3768	470	↓ 239
ENTROPY -BASED	Think-or-Not ($\eta = 0.1$)	34.0	↑ 8.0	203k	↓ 70k	4053	↓ 1400	284	↓ 53
	Think-or-Not ($\eta = 0.2$)	38.0	↑ 4.0	178k	↓ 45k	3560	↓ 907	242	↓ 11
	Think-or-Not ($\eta = 0.3$)	32.0	↑ 10.0	173k	↓ 40k	3451	↓ 798	220	↑ 11
	ARES (RL)	42.0	–	133k	–	2653	–	231	–

reasoning tokens further than ARES on *Airline*, no single threshold transfers across domains: DEER attains its best efficiency at $\delta = 0.50$ on *Retail* but at $\delta = 0.95$ on *Airline*, and Think-or-Not's best accuracy threshold flips between $\eta = 0.3$ (*Retail*) and $\eta = 0.2$ (*Airline*). This sensitivity to per-domain hyperparameters illustrates the limitation of static thresholding in multi-turn agent settings, where the optimal effort distribution depends on the trajectory dynamics rather than on a fixed token-level signal.

E Comparison with Learned Routers: Protocol and Router Overhead

Protocol. We evaluate TCAndon-Router [Zhao et al., 2026], Router-R1 [Zhang et al., 2025a], and ToolOrchestra [Su et al., 2025] with their released routing policies on `TAU-Bench` *Retail* and *Airline*. Every router is tested in two routing spaces: (i) *effort routing* over the low, medium, and high modes of `gpt-oss-20b`, and (ii) *model routing* over `Qwen3-4B`, `gpt-oss-20b`, and `Qwen3-32B`. In both spaces, every router receives the same number of explicitly described choices.

Token accounting. Table 3 reports two token metrics. T_{task} is the number of reasoning tokens per task generated by the selected agent models. O_{task} covers all output tokens of the selected agent models, reasoning and non-reasoning, converted into `gpt-oss-20b`-equivalent output tokens using each model's output-token price. For example, 2481 / 26221 for TCAndon with model routing on *Retail* denotes 2,481 reasoning tokens and an output cost equivalent to 26,221 `gpt-oss-20b` output tokens.

Router overhead. Table 6 decomposes the cost of the routers themselves. *Out-only* counts only the router's generated output tokens, whereas *In+out* also counts the input context the router processes at every call. The baseline routers re-read approximately 2.5k–2.9k input tokens per call and generate 87–465 output tokens, whereas the ARES router processes only ∼68–77 tokens per step (Table 8). In input-plus-output cost, the ARES router is 21–69× cheaper than the baseline routers on *Retail* and 16–58× cheaper on *Airline*.

F Heuristic Routing Baseline on WebArena

The temporal effort pattern visualized in Figure 3(b) suggests a simple non-learned alternative: sample the next-step effort from a step-conditioned categorical distribution that mimics ARES's aggregate behavior. Concretely, for steps 0 through 4 we draw effort from $(\text{low}, \text{medium}, \text{high}) =$

Table 6: Overhead of the routers themselves on **TAU-Bench**. *Out tok.*: router output tokens per task, converted into `gpt-oss-20b`-equivalent tokens by output price. *Out-only*: cost (USD) of the router's output tokens per run. *In+out*: cost (USD) of the router's input and output tokens per run.

Router	Size	Calls/task	Out tok.	Out-only ($)	In+out ($)
TAU-Bench *Retail*					
TCAndon	4.0B	14.2	1392	0.0049	0.0478
Router-R1	3.1B	13.5	1881	0.0066	0.0393
ToolOrchestra	8.2B	14.3	13453	0.0475	0.1309
ARES	1.7B	14.4	462	0.0019	0.0019
TAU-Bench *Airline*					
TCAndon	4.0B	10.9	1053	0.0016	0.0157
Router-R1	3.1B	10.4	1433	0.0022	0.0130
ToolOrchestra	8.2B	11.3	12000	0.0184	0.0464
ARES	1.7B	11.5	418	0.0008	0.0008

Table 7: Comparison with a step-conditioned categorical-distribution heuristic on **WebArena**. The heuristic samples low/medium/high effort with probabilities (50%, 5%, 45%) for steps $0-4$ and (35%, 5%, 60%) afterwards, mimicking the temporal pattern observed in Figure 3(b).

Method	Acc. (%)	Δ_{Acc}	T_{total}	T_{task}	T_{step}
Categorical Distribution	39.5	$\uparrow$ 7.0	1072k	8391	923
ARES	46.5	–	1512k	11723	1324

$(50\%, 5\%, 45\%)$, and after step 4 we switch to $(35\%, 5\%, 60\%)$. Table 7 reports the comparison on `WebArena`. The categorical heuristic recovers most of the token reduction, but loses 7.0 percentage points of accuracy relative to ARES, indicating that simply matching the marginal distribution of effort selections is insufficient: effective routing requires per-step, context-aware reasoning about the current observation rather than a fixed time-conditioned prior.

G Router Overhead Analysis

The router cost in our pipeline is trivial. We analyze it from three complementary angles: per-step token consumption, end-to-end wall-clock time, and monetary cost.

Per-step tokens and wall-clock time. Table 8 decomposes the per-step token usage into router and agent components, and reports the total wall-clock time (CT) to complete the entire benchmark. The 1.7B router introduces only $\sim$68–77 tokens per step, which is small compared to the agent's hundreds-to-thousands of reasoning tokens at high effort. To strictly isolate the router's standalone time overhead from interactions with the agent's stochastic generations, we additionally include an *Ares-replicate* setting that forces the environment to follow the exact action sequence chosen by ARES so that any difference comes purely from running the router. The marginal time cost from the router is only 8.5% (2438.3 s vs. 2246.1 s on *Retail*), while the agent's inference time is reduced by 30.8% relative to the high-effort baseline (3524.1 s).

Monetary cost. Using public OpenRouter pricing for `gpt-oss-20b` ($0.20 per million tokens) and a conservative parameter-proportional rate of $0.017 per million tokens for the 1.7B router, Table 9 reports the dollar cost of completing `TAU-Bench` *Retail*. The router accounts for less than 2% of ARES's total cost, and the overall financial cost is reduced by roughly 52.0% relative to the high-effort baseline.

Table 8: End-to-end cost decomposition on **TAU-Bench**, comparing the per-step token consumption of the agent (T_{agent}) and the router (T_{router}), and the total wall-clock time (CT) for completing the entire benchmark. The router (`Qwen3-1.7B`) is served on 1×H100 and the agent (`gpt-oss-20b`) on 2×H100 with tensor parallelism. *Ares-replicate* forces the environment to follow ARES's exact trajectory to isolate the router's standalone time overhead.

Setting	Acc. (%)	T_{router}	T_{agent}	CT (s)
TAU-Bench *Retail*				
Low	35.0	0	15	1668.7
High	54.8	0	634	3524.1
ARES	58.5	68	286	2438.3
ARES-replicate	58.5	68	286	2246.1
TAU-Bench *Airline*				
High	38.0	0	1654	2277.5
ARES	42.0	77	231	–

Table 9: Monetary cost (USD) of completing **TAU-Bench** *Retail*. The router cost is conservatively estimated as proportional to the parameter scale ($0.017 per million tokens for the 1.7B router, against $0.20 per million tokens for the 20B agent based on public OpenRouter pricing).

Setting	Agent ($)	Router ($)	Total ($)
High	0.2014	–	0.2014
ARES	0.0947	0.0019	0.0966

H Cross-Scale and Cross-Architecture Generalization

To assess the robustness of ARES, we study generalization along three orthogonal axes: (i) scaling the agent backbone within the same model family, (ii) transferring to a different reasoning architecture with hybrid thinking modes, and (iii) varying the router's own size. All evaluations in this section are conducted on `TAU-Bench` *Retail*.

H.1 Scaling Up to a 120B Backbone

A natural test of generalization is whether the ARES router transfers to a much larger backbone agent. Since the router was originally trained solely on the significantly smaller `gpt-oss-20b`, this experiment probes the framework's robustness to a substantial shift in the underlying agent's capability. We compare two routing strategies for `gpt-oss-120b`: (a) *Transfer* — applying the original router trained on `gpt-oss-20b` zero-shot to the 120B agent; and (b) *Dedicated* — freshly synthesizing data from `gpt-oss-120b` and training a new 1.7B router end-to-end.

As shown in Table 10, despite the backbone being six times larger than the training source, the transferred router still achieves 65.2% accuracy and recovers the majority of the high-effort performance (67.8%) while reducing token consumption by approximately 23%. The dedicated router pushes performance further beyond the high-effort ceiling (69.6%) and cuts reasoning tokens by roughly half. Together, these results suggest that the reasoning rationales and effort-selection cues learned by ARES are largely scale-invariant, and that targeted retraining unlocks additional gains.

H.2 Cross-Architecture Pipeline Generalization and the Effect of Router Scale

To test that the ARES pipeline is not specific to discrete `low/medium/high` effort modes, we apply it to the `Qwen3` family [Yang et al., 2025a], whose native "hybrid thinking" interface exposes only two modes: *think* and *no-think*. We re-run the entire data-generation pipeline for `Qwen3-8B` and `Qwen3-32B` and train a per-backbone router. To additionally probe the impact of the router's own capacity, we train an 8B router (`Qwen3-8B`) and pair both router sizes with three agent backbones

Table 10: Cross-scale generalization to a much larger backbone agent (`gpt-oss-120b`) on `TAU-Bench` *Retail*. *Transfer*: the ARES router is trained on `gpt-oss-20b` and applied zero-shot to the 120B agent. *Dedicated*: a fresh 1.7B router is trained end-to-end with data synthesized from the 120B agent.

Setting	**Acc. (%)**	Δ_{Acc}	T_{total}	T_{task}	T_{step}
Low	49.4	↑ 20.2	26k	231	17
Medium	62.0	↑ 7.6	111k	972	74
High	67.8	↑ 1.8	558k	4856	380
ARES (Transfer)	65.2	↑ 4.4	428k	3729	293
ARES (Dedicated)	69.6	–	298k	2591	204

Table 11: Generalization across reasoning architectures and router scales on `TAU-Bench` *Retail*. We pair 1.7B and 8B routers with three agent backbones spanning two model families. For `Qwen3` agents, the discrete reasoning levels correspond to the model's native *think*/*no-think* hybrid thinking modes.

Setting	**Acc. (%)**	Δ_{Acc}	T_{total}	T_{task}	T_{step}
Agent: `Qwen3-8B`					
No Think	33.9	↑ 15.7	–	–	–
Think	47.8	↑ 1.8	782k	6804	479
Random	40.9	↑ 8.7	419k	3643	245
ARES (1.7B)	50.4	↓ 0.8	519k	4510	301
ARES (8B)	49.6	–	500k	4344	278
Agent: `Qwen3-32B`					
No Think	47.0	↑ 13.9	–	–	–
Think	55.7	↑ 5.2	509k	4425	351
Random	53.0	↑ 7.9	303k	2631	183
ARES (1.7B)	56.5	↑ 4.4	318k	2765	207
ARES (8B)	60.9	–	340k	2959	209
Agent: `gpt-oss-20b`					
Low	35.0	↑ 20.7	25k	223	15
Medium	47.3	↑ 8.4	137k	1198	82
High	54.8	↑ 0.9	1007k	8756	634
Random	43.5	↑ 12.2	359k	3126	215
ARES (1.7B)	54.8	↑ 0.9	652k	5677	388
ARES (8B)	55.7	–	659k	5737	387

spanning two model families: `Qwen3-8B`, `Qwen3-32B`, and `gpt-oss-20b`. Table 11 summarizes the results, from which three trends emerge:

- **The ARES pipeline transfers cleanly to hybrid thinking modes.** On both `Qwen3-8B` and `Qwen3-32B`, ARES matches or slightly exceeds the always-*think* configuration in accuracy while reducing T_{total} by roughly 30%–40%, despite the action space collapsing from three to two reasoning levels.
- **A larger router yields better routing on more capable agents.** On `Qwen3-32B` and `gpt-oss-20b`, replacing the 1.7B router with the 8B router yields a clear accuracy bump (e.g., $+4.4\%$ on `Qwen3-32B`), whereas the gain on the smaller `Qwen3-8B` agent is negligible. We attribute this to two factors: smaller agents are more likely to hit an inherent capability ceiling that no routing strategy can lift, and their higher action-prediction variance under the same effort level makes the deterministic-mapping assumption (Appendix J.5) less reliable during data construction.
- **No special advantage from family alignment.** The router observes only the agent's external actions and the environment state, never the agent's internal chain-of-thought. Consequently, sharing the router's and agent's model family does not provide a measurable benefit over crossing families.

Table 12: Agreement between human annotators and the LLM judge on the reasoning-effort labeling task. For each benchmark, we randomly sample 50 step-level decisions and have a human re-annotate the action-equivalence judgment that the LLM judge produced.

Benchmark	Human vs. LLM Judge Agreement
TAU-Bench	50 / 50 (100%)
WebArena	50 / 50 (100%)
BrowseComp-Plus	48 / 50 (96%)

I LLM-as-Judge Reliability

The reasoning-effort annotation pipeline (Section 3.2, Phase 2) and the deep-research evaluation in `BrowseComp-Plus` both rely on an LLM judge. The judge's task in our pipeline is well-defined: it is asked only to decide whether two candidate actions are functionally equivalent, which for tool-use and web agents reduces to comparing the tool name and the key arguments (*e.g.*, `click[1213]` vs. `click[1234]`).

To verify reliability empirically, we randomly sample 50 step-level annotations from each of the three benchmarks and have a human annotator independently re-judge action equivalence. Table 12 reports the agreement rates. The judge agrees with the human annotator on every sample for `TAU-Bench` and `WebArena`, and on $48/50$ samples for `BrowseComp-Plus`. Both disagreements on `BrowseComp-Plus` arose from `search` actions whose query strings were lexically distinct but returned identical retrieved documents; the human annotator treated these as matches, while the judge — which evaluates actions offline without access to runtime retrieval feedback — flagged the lexical mismatch and conservatively labeled them as non-matching. This is a strict but expected behavior under the offline annotation setting.

Answer judge on `BrowseComp-Plus`. For `BrowseComp-Plus` evaluation we use `GPT-4o` as the answer judge, as specified by the official implementation. The judge receives only the question, the reference answer, and the prediction, and returns a binary judgment. We re-judge the identical ARES predictions with `GPT-5.5` at high reasoning effort. `GPT-4o` marks 62 of 150 predictions correct (41.3%) and `GPT-5.5` marks 64 (42.7%); the two judges agree on 148 of 150 predictions (98.7%, Cohen's $\kappa = 0.973$). In both disagreements, `GPT-5.5` accepts an answer that is semantically equivalent or more specific than the reference, such as a store-name abbreviation or an answer that additionally includes the city and country.

J Supervised Fine-tuning Pipeline: Detailed Construction and Theoretical Analysis

This appendix complements Section 3.2 by providing (i) the formal connection between the step-wise SFT label and the trajectory-level objective from Section 3.1, (ii) the full data-construction details we summarized in the main paper, and (iii) a discussion of the residual gap that motivates the reinforcement-learning stage in Section 3.2.

J.1 Trajectory-level Objective and Step-wise Surrogate

Restating the cost-augmented objective from Section 3.1, we seek a router policy π_θ that maximizes

$$J_{\text{traj}}(\theta) \;=\; \mathbb{E}_{\tau\sim\mathcal{T}(\theta,\phi)}\Big[\, R(\tau) \;-\; \lambda \sum_{t=1}^{T} c(e_t) \,\Big], \tag{5}$$

where $R(\tau) = \mathcal{V}(\tau)$ and $c(e_t)$ is the reasoning cost at step t. Direct optimization of (5) is intractable because the search space scales as $|\mathcal{E}|^T$ and changing e_t at any single step shifts the entire downstream trajectory.

Conditioned on a fixed reference trajectory $\tau^* = (o_1, a_1^*, \ldots, o_T, a_T^*)$, our pipeline instead solves a per-step problem,

$$e_t^* \;=\; \arg\min_{e\in\mathcal{E}} \; c(e) \quad \text{s.t.} \quad a_t(e, r_t) = a_t^*, \tag{6}$$

i.e., the lowest-cost effort that still reproduces the reference action at step t. Sections J.2–J.4 below detail how each phase of our pipeline materializes (6), while Section J.5 discusses the conditions under which the surrogate tightly bounds (5).

J.2 Phase 1: Trajectory Collection (Details)

Rather than searching for efficiency across arbitrary action sequences, we anchor our analysis on high-quality, successful trajectories. For each training task x, we sample N successful trajectories using the agent $\mathcal{M}_{\text{agent}}$ under the maximum effort level e_{high}, and select the most concise trajectory τ^* as the reference path. The rationale for this selection is two-fold. First, trajectories with more steps inflate the total reasoning cost and make it harder to identify the true minimum effort required for the core task; concise trajectories provide a cleaner signal of the essential reasoning needed. Second, by fixing a high-quality action sequence, we transform the long-horizon optimization in (5) into a series of independent step-wise labeling tasks, allowing us to precisely measure how different reasoning levels affect the execution of each ground-truth action a_t^* in isolation. The shortest successful trajectory is one choice of reference; other criteria, such as the successful trajectory with the lowest reasoning-token cost, fit the same pipeline unchanged. Because the RL stage optimizes success and realized cost without a reference, this choice mainly affects the SFT initialization.

We also note that the ARES framework is compatible with external data sources. The router can be trained using existing open-source agent datasets, even if those trajectories were generated by a different LLM. This flexibility allows ARES to leverage high-quality interaction data without always requiring expensive self-sampling. As demonstrated in our experiments (Section 4), the router generalizes effectively across different interaction styles and model sources.

J.3 Phase 2: Reasoning Effort Annotation (Details)

With the optimal trajectory τ^* as a reference, we decompose it into individual decision steps to identify the minimum required reasoning effort for each action. To account for the inherent stochasticity of LLMs, we employ a multi-trial verification process. For each effort level $e \in \{e_{\text{low}}, e_{\text{mid}}, e_{\text{high}}\}$, we sample the agent's response K times ($K = 3$ in our experiments) and compare the predicted actions $\{\hat{a}_{t,k}^{(e)}\}_{k=1}^{K}$ against the ground truth a_t^*. We define a verification function $\mathcal{V}(\hat{a}, a_t^*)$ that returns 1 if the predicted action is functionally equivalent to the ground truth and 0 otherwise. We require the action to be correct in at least M out of K trials. The candidate set of sufficient levels $\mathcal{C}_t$ is then defined as

$$\mathcal{C}_t = \left\{ e \in \mathcal{E} \;\middle|\; \sum_{k=1}^{K} \mathcal{V}\big(\hat{a}_{t,k}^{(e)}, a_t^*\big) \geq M \right\}. \tag{7}$$

To make $\mathcal{V}$ both rigorous and practical, we tailor the equivalence criterion to each agent domain. Two actions are considered functionally equivalent if they produce the same effect on the environment:

- **Tool-use agents.** An action is valid if the predicted tool name matches a_t^* and the key parameters are identical.
- **Web agents.** An action is correct if it interacts with the environment in exactly the same manner as a_t^* (*e.g.*, clicking a specific element such as `click[1316]`).
- **Deep research agents.** The primary action is search; we use an LLM judge to determine if two search queries are semantically equivalent.

Across all three agent types, when the agent outputs natural language messages (*e.g.*, soliciting user information or providing a final answer), we additionally employ an LLM judge to verify semantic consistency.

Finally, we assign the label $y_t = \arg\min_{e \in \mathcal{C}_t} c(e)$, *i.e.*, the lowest-cost sufficient level. For example, if both e_{low} and e_{mid} are sufficient, we label the step as e_{low} to maximize efficiency. The fallback and discard rules are mutually exclusive. If $\mathcal{C}_t = \emptyset$ but at least one sample at some effort level matches a_t^*, we fall back to the lowest effort level with the highest non-zero match rate. Only if no sample at any effort level matches a_t^* is the step discarded. The discard rate is 26.5% on `TAU-Bench`, 19.9% on `BrowseComp-Plus`, and 52.2% on `WebArena`, where reference actions come from trajectories released by AGENTOCCAM and must be reproduced exactly at the element level. Discarded steps

reduce supervision coverage, most on `WebArena`. Coverage can be raised by collecting reference trajectories with the backbone agent itself, as we do for `BrowseComp-Plus`, or by matching actions on the resulting page state instead of the exact element ID. This exhaustive labeling defines a per-step computational lower bound that the router learns to recover.

Multiple valid actions. How often a step admits several equally valid next actions depends on the task type. In `TAU-Bench` and `WebArena`, the reference plan usually admits a single task-consistent next action, namely a specific tool call with its key arguments or a specific element interaction, so we match actions exactly. `BrowseComp-Plus` permits reordered or reformulated searches, so the judge accepts semantically equivalent queries. The SFT labels do not need to enumerate every valid action: SFT initializes the router, and the RL stage (Section J.6) optimizes final success and realized reasoning cost from on-policy rollouts without reference actions, which lets the router explore alternative actions and trajectories.

J.4 Phase 3: Rationale Generation (Details)

To enhance the router's predictive accuracy, we train it to emit a brief rationale r_t before producing the final effort label. The rationale forces the lightweight router to analyze the current state explicitly rather than perform a direct mapping from context to label; we ablate its contribution in Section 4.6.

To synthesize high-quality rationale data, we employ a powerful teacher model to perform post-hoc analysis on the labeled trajectories. For each step t, the teacher receives the interaction history h_t, the current observation o_t, and the verified label y_t, and is tasked with justifying why y_t is the most appropriate level for the upcoming action. The generated rationale integrates three factors: the complexity of the latest observation, the current progress of the overall task, and the estimated difficulty of the required sub-task. To ensure that the router's own reasoning does not introduce significant latency, we impose a strict length constraint on the teacher, limiting the rationale to a concise 3–5-sentence summary. This keeps the router lightweight and fast, providing a favorable trade-off between its own token overhead and the substantial token savings it enables for the agent.

J.5 When the Surrogate Bounds the Trajectory Optimum

Optimizing (6) at every step is equivalent to optimizing (5) provided two conditions hold:

- **(C1) Deterministic effort-to-action mapping.** The agent's action a_t is conditioned on a stochastic rationale $r_t \sim P(r \mid o_{\leq t}, e_t)$, so a low-effort setting may produce a_t^* by chance. The multi-trial check in (7) (Phase 2) accepts e only if the candidate action matches a_t^* in M out of K independent samples, driving the marginal $P(a_t \mid o_{\leq t}, e_t)$ to peak reliably at a_t^* and ensuring that the per-step minimizer is stable rather than a stochastic accident.
- **(C2) Reference trajectory is globally optimal.** If τ^* is itself sub-optimal, step-wise minimization can only converge to a sub-optimal trajectory. Phase 1 samples multiple successful trajectories per task with the maximum-effort agent and keeps the *shortest* one, which removes loops, failed actions, and redundant recovery steps from the reference. The RL stage (Section J.6) then optimizes (5) directly, without a reference trajectory.

When (C1) and (C2) both hold, the per-step solution to (6) is the projection of the trajectory-level minimizer of (5) onto step t, so the supervised signal upper-bounds the trajectory cost achievable by any router policy that respects the reference trajectory.

J.6 Closing the Residual Gap with RL

Conditions (C1)–(C2) are sufficient but not necessary, and they tighten only when the reference trajectory is unique up to redundant actions. In domains where multiple distinct trajectories are equally optimal, or where the agent's action distribution remains broad even after multi-trial verification, the surrogate becomes loose. The reinforcement-learning stage of ARES directly optimizes (5) on rollouts generated under the inference distribution, providing the trajectory-level credit assignment that the step-wise surrogate cannot supply, and empirically yields further gains on top of the SFT-initialized router (see Table 2).

Table 13: One-time offline cost of constructing the ARES training data with proprietary teacher models. Pricing reflects the rates at the time of data collection ($2.5 / 1M input tokens for `GPT-4o`, $1.25 / 1M input tokens for `GPT-5`, $10 / 1M output tokens for both).

Stage	Input Tokens	Output Tokens	Cost (USD)
LLM Judge (`GPT-4o`)	46.4M	0.6M	122.0
Rationale Generation (`GPT-5`)	102.8M	3.5M	163.5
Total	**149.2M**	**4.1M**	**285.5**

K KV Cache Reuse Across Reasoning Modes

A practical advantage of ARES over heterogeneous model routing is that it operates entirely within a single backbone model and therefore preserves the KV cache across reasoning-mode switches. Switching between models of different sizes invalidates the accumulated KV cache, since the source model's hidden representations are not consumable by the destination model. By contrast, modern reasoning LLMs already expose per-turn reasoning controls inside the same model. As a concrete example, the `Qwen3` series toggles its reasoning behavior simply by appending the special tags `/think` or `/no_think` to the current user message, without any change to model weights or context layout:

Per-turn reasoning control with Qwen3

```
# Turn 1:  thinking enabled by default
r1 = chatbot.generate_response("How many r's in strawberries?")
# Turn 2:  disable thinking via /no_think
r2 = chatbot.generate_response("Then, how many r's in blueberries?
/no_think")
# Turn 3:  re-enable thinking via /think
r3 = chatbot.generate_response("Really?  /think")
```

When a subsequent turn arrives, regardless of which tag is attached to it, the entire prefix from the system prompt down to `r3` remains byte-identical, so the inference engine reuses the cached keys and values for the shared prefix. The same property holds for the discrete `reasoning_effort` parameter exposed by `gpt-oss`-style models: the per-turn effort selection only modulates the model's decoding behavior on the new turn, and never alters the historical context. Consequently, the only KV-cache cost introduced by ARES is the small footprint of the router's own forward pass over the shared trajectory representation, which we already account for in Appendix G.

L Data Annotation Cost

Our pipeline incurs a one-time, offline cost on proprietary teacher models for action-equivalence judgment and rationale generation during dataset construction. Table 13 reports the exact token usage and dollar cost for the full ARES dataset.

The total cost of approximately $285 is well within standard research budgets. Because the rationale generator is grounded in verified execution traces and observed environment feedback (Section 3.2, Phase 3), the resulting supervision is constrained to factual descriptions of the trajectory, which empirically bounds the bias inherited from the teacher model.

M Reward Normalization Ablation: Training Dynamics

This appendix expands the reward-design ablation summarized in Section 4.6. We compare the trajectory-averaged cost reward used by ARES, $R_{\text{cost}} = \frac{1}{T}\sum_{t=1}^{T} c(e_t)$, against an accumulated alternative $R_{\text{cost}} = \sum_{t=1}^{T} c(e_t)$. To keep successful trajectories net-positive under the accumulated

form, we proportionally scale the per-step penalties to $c(e_t) = -0.02, -0.06, -0.12$ for low, medium, and high effort, respectively.

Figure 4(c) tracks the GRPO training dynamics on `TAU-Bench` *Airline*. The right sub-panel shows that with normalization, high-effort usage is driven down aggressively to $\sim 15\%$, while without normalization it plateaus near 30%. The left sub-panel shows a complementary effect on the medium effort: since medium is the empirically optimal level for *Airline*, the normalized reward steadily increases its usage throughout training, whereas the unnormalized variant initially raises medium-effort usage but then drives it back down after roughly 80 training steps as the cumulative penalty over long trajectories outweighs the outcome reward.

These training-dynamic differences propagate to the final evaluation in Figure 4(b): the normalized reward achieves both higher task accuracy and approximately 15% fewer total reasoning tokens (133k vs. 157k). The accumulated penalty fails to balance per-step costs against the final outcome — long trajectories accumulate disproportionate penalty even when each individual step is well allocated — preventing convergence to the most efficient and effective effort distribution.

N Limitations

Our evaluation focuses on text-based tool-use, deep-research, and web-navigation agents. Extending ARES to multi-modal agents (*e.g.*, vision-language web or GUI agents) and infra-level optimizations are left to future work.

O Broader Impacts

By substantially reducing the reasoning-token cost of LLM agents at near-equal accuracy, ARES lowers the monetary, latency, and energy footprint of agent deployment, which broadens accessibility of capable agents. ARES introduces no new agent capability beyond what the underlying backbone already exposes; it only modulates a per-turn reasoning level, so we do not foresee additional negative societal impacts beyond those already associated with the underlying LLM agents.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state our three contributions (per-step adaptive reasoning effort selection, the SFT+RL training pipeline, and evaluation across three agent benchmarks) and the headline 52.7% reasoning-token reduction, all of which are substantiated in Sections 4 and Appendices H–G.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix N discusses the limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formally stated theorems; the only theoretical content is the informal sufficiency argument in Appendix J.5 (conditions C1, C2), whose assumptions are explicitly stated alongside the argument.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3.2 describes the SFT pipeline and Section 3.3 describes the RL setup; Appendix C provides environment details, agent backbones, training data sources, the annotation pipeline configuration (K=3, M=3, GPT-4o judge, GPT-5 teacher), the RL prompt-filter configuration (Appendix C.5), and the full SFT/RL hyperparameters; Appendix A lists all router/annotator/rationale prompts and Appendix B gives a worked training example.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and data are available at `https://github.com/UCSB-NLP-Chang/Ares`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 reports benchmarks, backbones, training data sources, evaluation metrics, and baseline families; Appendix C (Training Hyperparameters) lists optimizer, learning rates, batch sizes, epochs, warmup, weight decay, GRPO group size, KL coefficient, and prompt/response length caps for both SFT and RL, and Appendix C.5 details data-filtering thresholds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The experiments on `BrowseComp-Plus` and `WebArena` are evaluated using greedy decoding to ensure deterministic results. Since `TAU-Bench` introduces an additional user simulator model, which adds variance, the `TAU-Bench` evaluation results are averaged over three independent trials.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix G reports the end-to-end wall-clock time of each evaluated configuration on `TAU-Bench` *Retail* (Table 8) and the dollar cost of each setting under public OpenRouter pricing (Table 9); Appendix L additionally reports the one-time data-annotation cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix O discusses both positive impacts (reduced inference cost, latency, and energy footprint, broadening accessibility of reasoning agents) and potential negative impacts (cheaper agents lowering the barrier to large-scale autonomous deployment), together with the mitigation that ARES introduces no new agent capability beyond what the underlying model already exposes.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifact is a 1.7B router that only predicts a discrete reasoning-effort level for an external agent; it carries no high-risk-misuse profile that would require gated release.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used assets are cited in-text: benchmarks `TAU-Bench` [Yao et al., 2024], `BrowseComp-Plus` [Chen et al., 2025, Wei et al., 2025], `WebArena` [Zhou et al., 2023]; backbones `gpt-oss-20b/120b`, `Qwen3-8B/32B` [Yang et al., 2025a]; framework AGENTOCCAM [Yang et al., 2024]; trajectory data `APIGen-MT` [Prabhakar et al., 2025]; we use each asset within its publicly-released terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets (the ARES effort-annotated SFT dataset, the rationale-augmented training data, and the trained router checkpoints) are documented end-to-end in Section 3.2 and Appendices J, C.5, L, and A–B (full prompts and a worked example); they will be released with documentation and license upon publication.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The work does not involve crowdsourcing or research with human subjects; the only human-in-the-loop component is a single authors' spot-check of 50 LLM-judge annotations per benchmark (Appendix I).

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The work does not involve human-subjects research, so no IRB approval is required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are core to the methodology and are described accordingly: `GPT-4o` is the action-equivalence judge in the annotation pipeline, `GPT-5` is the rationale-generation teacher, `GPT-4o` is the user simulator on `TAU-Bench`, and the routed agent backbones are `gpt-oss-20b/120b` and `Qwen3-8B/32B`; usage and costs are detailed in Appendices C and L.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.